ADAPTION BIOTECHNOLOGISCHER PROZESSE FÜR DIE INTERAKTIVE UMSETZUNG IN DIGITAL UNTERSTÜTZTER LABORUMGEBUNG, ENTWICKLUNG UND EVALUATION

Von der Naturwissenschaftlichen Fakultät der
Gottfried Wilhelm Leibniz Universität Hannover

zur Erlangung des Grades

DOKTOR DER NATURWISSENSCHAFTEN
(DR. RER. NAT.)

genehmigte Dissertation
von

Marc Julian Porr, M. Sc.

2021

REFERENT:	Prof. Dr. rer. nat. Thomas Scheper
KORREFERENT:	apl. Prof. Dr. rer. nat. Sascha Beutel
TAG DER PROMOTION:	16.04.2021

„LERNEN IST WIE RUDERN GEGEN DEN STROM. HÖRT MAN DAMIT AUF, TREIBT MAN ZURÜCK.“

LAOZI

DANKSAGUNG

Ich möchte mich ganz herzlich bei allen Kollegen – aktuellen sowie ehemaligen – des Instituts für Technische Chemie (TCI) der Leibniz Universität Hannover und allen anderen Menschen bedanken, die mich in der Zeit meiner Promotion begleitet und unterstützt haben.

Besonders danke ich meinem Doktorvater, Prof. Dr. Thomas Scheper, und meinem Arbeitskreisleiter PD Dr. Sascha Beutel für die herausragende Betreuung sowie allen Mitgliedern der Prüfungskommission. Vielen herzlichen Dank für die Möglichkeit am TCI zu promovieren und die schöne Zeit hier!

Allen Kollegen, mit denen ich während meiner Zeit am TCI das Büro geteilt habe, danke ich besonders für die (fachlichen) Gespräche und den vielen Spaß, den wir gemeinsam hatten. Besonders danke ich Dr. Daniel Marquard und Ferdinand Lange, die mir in dieser Zeit immer eine große Hilfe waren. Außerdem möchte ich allen studentischen MitarbeiterInnen und AbschlussstudentInnen danken, die mich in dieser Zeit begleitet haben: Henrike Heinemann, Laura Niemeyer, Sebastian Schwarz, Jonas Bock, Michelle Djuari, Tessa Habich und Karolina Sikora.

Mein besonderer Dank gilt ebenfalls Thorleif Hentrop, Kai Patzer, Thorsten Strempel, Friedbert Gellermann, Jonas Uhlendorf, Martina Weiß und Ulrike Dreschel. Ohne euch wäre vieles nicht möglich gewesen.

Zuletzt danke ich ganz besonders meiner Familie und meiner Freundin. Jana, du gibst mir immer Rückhalt und Unterstützung in allem, was ich schaffen möchte. Danke euch allen!

KURZFASSUNG

Titel: Adaption biotechnologischer Prozesse für die interaktive Umsetzung in digital unterstützter Laborumgebung, Entwicklung und Evaluation

Schlagwörter: digitale Transformation, Labor 4.0, System-Architektur, SiLA2, Digitalisierung, Mensch-Computer-Interaktion, Geräteintegration, biotechnologisches Labor, digitale Nutzerinteraktion

Die vorliegende Arbeit beschäftigt sich mit der digitalen Transformation des biotechnologischen Labors. Ziel der Bemühungen ist, einen Mehrwert sowohl in der Qualität der erzeugten Ergebnisse, als auch in den Arbeitsabläufen für den Wissenschaftler oder Laboranten zu schaffen. Besonderes Augenmerk liegt dabei auf der Interaktion des digitalisierten Laborsystems mit dem Benutzer.

Es wird beschrieben, welche Maßnahmen notwendig sind, um ein solches Laborsystem aufbauen und betreiben zu können. Neben den Hardware-Voraussetzungen für den Betrieb wird insbesondere auf die standardisierte Anbindung von Laborgeräten eingegangen. Da momentan noch kein allgemeiner Kommunikationsstandard in diesem Bereich existiert, werden Mittel und Wege aufgezeigt, wie eine umfassende Integration trotzdem möglich ist. Dazu wurde u. a. ein Hardware-Modul zur Anbindung von Alt- oder Bestandsgeräten entwickelt und evaluiert.

Um die Interaktion des Menschen mit dem Laborsystem so unkompliziert wie möglich zu gestalten, werden verschiedene Formen der Nutzerinteraktion (Sprachsteuerung, *Head-Mounted-Displays*, etc.) untersucht und vorgestellt. Das gesamte System ist so ausgelegt, dass die Steuerung von zentraler Stelle aus möglich ist und alle Daten in einem Laborserver zusammenfließen. Diese Architektur schafft die Voraussetzungen für eine generische und unkomplizierte Anbindung verschiedenster Nutzerinteraktionsgeräte.

Im Rahmen der Arbeiten wurde weiterhin ein Prozessleitsystem entwickelt, dass die Formulierung und Durchführung von Arbeitsabläufen im digitalisierten Labor einfach und intuitiv ermöglicht. Dabei sollen Implementierungsdetails und konkrete Probleme der Geräteanbindung abstrahiert werden und die Abläufe möglichst unabhängig von den konkreten Modellen der Geräte werden, die zu ihrer Durchführung notwendig sind.

Durch die zentrale Geräteanbindung und Datenverarbeitung werden Möglichkeiten für Abläufe geschaffen, die die FAIR (*Findable, Accessible, Interoperable, Reusable*)-Richtlinien zum Umgang mit wissenschaftlichen Daten einhalten. Das digitalisierte Labor wurde für die Durchführung von Beispielabläufen und Nutzerinteraktionsstudien verwendet und steht voll funktional zur Verfügung. Die Software und Systemarchitekturen wurden in verschiedenen wissenschaftlichen Veröffentlichungen dokumentiert und viele Komponenten sind unter offenen Lizenzen verfügbar.

ABSTRACT

Title: Adaption of Biotechnological Processes for the Interactive Execution in a Digitally Assisted Laboratory Environment, Development and Evaluation

Keywords: digital Transformation, Lab 4.0, System Architecture, SiLA2, Digitization, Human-Computer-Interaction, Device Integration, biotechnological Lab, digital User Interaction

This thesis deals with the digital transformation of the biotechnological laboratory. The goal of the efforts is to create benefit for the scientist or laboratory technician, both in the quality of the results produced and in the work processes. Special attention is paid to the interaction of the digitized laboratory system with the user.

It is described which measures are necessary to set up and operate such a laboratory system. In addition to the hardware requirements for operation, the standardized connection of laboratory devices is discussed in particular. Since there is currently no general communication standard in this area, ways and means are shown how a comprehensive integration is still possible. For this purpose, a hardware module for the connection of old or existing devices was developed and evaluated.

In order to make human interaction with the laboratory system as uncomplicated as possible, different forms of user interaction (voice control, head-mounted displays, etc.) are presented and evaluated. The entire system is designed in such a way that the control is possible from a central location and all data are merged in a laboratory server. This architecture creates the conditions for a generic and uncomplicated connection of various user interaction devices.

Within the scope of the work, also a process control system was developed enabling the simple and intuitive formulation and execution of workflows in the digitalized laboratory. Implementation details and specific problems of device connectivity are abstracted from the processes themselves, so that they become as independent as possible from the specific types of devices necessary for their execution.

The central device connection and data processing leverages the creation of processes that comply with the FAIR (Findable, Accessible, Interoperable, Reusable) guidelines for handling scientific data. The digitized laboratory has been used for the execution of sample procedures and user interaction studies and is fully functional. The software and system architectures have been documented in various scientific publications and many components are available under open licenses.

INHALTSVERZEICHNIS

ABKÜRZUNGSVERZEICHNISS

alphabetisch

ABKÜRZUNG	BEDEUTUNG
ADF	*Allotrope Data Format*
AnIML	*Analytical Information Markup Language*
API	*Application Programming Interface*
ARM	*Advanced RISC Machines*
ASTM	*früher: American Society for Testing and Materials*
ATDD	*AniML Technique Definition Document*
CC BY-SA 4.0	*Creative Commons Attribution-ShareAlike 4.0*
CERN	*Europäisches Forschungsinstitut für Kernphysik*
COM	*Component Model*
CSMA/CD	*Carrier Sense Multiple Access/Collision Detection*
DCOM	*Distributed Component Model*
DFG	*Deutsche Forschungsgemeinschaft*
DTD	*Document Type Definition*
eMMC	*embedded MultiMediaCard Connector*
FAIR	*Findable, Accessible, Interoperable, Reusable*
FBK	*Feldbuskoppler*
gRPC	*Rekursives Akronym: gRPC Remote Procedure Calls*
GxP	*Good Manufacturing/Laboratory Practice*
HDF 5	*Hierarchical Data Format 5*
HTML	*Hypertext Markup Language*
HTTP	*Hypertext Transfer Protocol*
IEEE	*Institute of Electrical and Electronics Engineers*
IIoT	*Industrial Internet of Things*
IoT	*Internet of Things*
IP	*Internet Protocol*
ISOC	*Internet Society*
IT	*Informationstechnik*
js	*JavaScript*
JSON	*JavaScript Object Notation*
LADS	*Laboratory Agnostic Device Standard*
LAN	*Local Area Network*
LIMS	*Labor-Informations- und Management-Systeme*
MPEG	*Moving Picture Experts Group*
NFDI	*Nationalen Forschungsdateninfrastruktur*
NOAA	*National Oceanic and Atmospheric Administration*
OD_{600}	*Optische Dichte bei 600 nm*
OPC	*Open Platform Communications*

OPC AE	*Open Platform Communications Alarms and Events*
OPC DA	*Open Platform Communications Data Access*
OPC HDA	*Open Platform Communications Historical Data Access*
OPC UA	*Open Platform Communications Unified Architecture*
PC	*Personal Computer*
PCCC	*Periodic Counter Current Chromatographie*
PCS	*PCCC-Kontrollsystem*
PHP	*rekursives Akronym: PHP Hypertext Preprocessor*
PMP	*PCCC-Hauptprogramm*
REST	*Representational State Transfer*
RFCs	*Request for Comment*
RISC	*Reduced Instruction Set Computer*
RJ-45	*Registered Jack 45*
RPC	*Remote Procedure Call*
SBC	*Single Board Computer*
SDK	*Software Development Kit*
SD	*Secure Digital Memory*
SiLA2	*Standardization in Lab Automation 2*
SOAP	*früher: Simple Object Access Protocol*
SOP	*Standard Operating Procedure*
SSH	*Secure Shell Protocol*
SSL	*Secure Sockets Layer*
SVG	*Scalable Vector Graphics*
TCP	*Transmission Control Protocol*
TCP/IP	*Transmission Control Protocol/Internet Protocol*
TLS	*Transport Layer Security*
UDP	*User Datagram Protocol*
USA	*Vereinigte Staaten von Amerika*
USB	*Universal Serial Bus*
UV/Vis	*Ultraviolett/Visible*
VM	*Virtual Machine*
W3C	*World Wide Web Consortium*
WLAN	*Wireless LAN*
www	*World Wide Web*
XHTML	*Extensible Hypertext Markup Language*
XML	*Extensible Markup Language*
XSD	*XML Schema Definition*
YAML	*rekursives Akronym: YAML Ain't Markup Language*

1 EINLEITUNG UND ZIELSETZUNG

In der vorliegenden Arbeit wird die Entwicklung und Umsetzung eines Digitalisierungskonzepts für biotechnologische Laboratorien gezeigt. Dabei wird besonderen Wert auf die Interaktion des digitalisierten Labors mit den darin arbeitenden Menschen gelegt.

1.1 EINLEITUNG

Die vierte industrielle Revolution ist in vollem Gange und viele Arbeitsabläufe und Mitarbeiter in der produzierenden Industrie profitieren bereits täglich von der Digitalisierung. So unterstützen Datenbrillen Mitarbeiter bei Wartungsarbeiten oder dem Kommissionieren. Datenbanken werden bereits seit geraumer Zeit zur strukturierten Erfassung von Arbeits- und Verbrauchsmitteln eingesetzt und die Interaktion der arbeitenden Menschen mit diesen Systemen wird kontinuierlich besser und intuitiver.

Der Konsument erwartet heute nahezu selbstverständlich, dass ein moderner Fernseher dazu in der Lage ist, Informationen mit anderen Geräten, wie der Beleuchtungsanlage oder dem Mobiltelefon, auszutauschen. Bei Laborgeräten hingegen wird das Fehlen solcher Möglichkeiten jedoch weitgehend akzeptiert und „Umwege“ – wie das papierbasierte Notieren von Messwerten zur späteren Übertragung in den Computer – werden stillschweigend in Kauf genommen.

Im Laboralltag herrschen analoge Methoden der Prozessführung und Dokumentation vor. Besonders Forschungslaboratorien, in denen viele Kleingeräte durch Mitarbeiter in manueller Arbeit bestückt und bedient werden, hinken im Bereich Digitalisierung der Industrie um Jahre hinterher. Durch lückenhafte Dokumentation und Fehler bei der manuellen Übertragung von Ergebnissen werden nicht nur Forschungsergebnisse gefährdet. Auch sind so Meta-Analysen von Ergebnissen mit Hilfe von Big-Data-Ansätzen kaum umsetzbar.

Dass mit Hilfe heute verfügbarer Technologien eine umfassende Digitalisierung von Umfeldern, in denen der Mensch mit verschiedenen Geräten interagiert, möglich ist, zeigt die Verbreitung des „Internets der Dinge“ (engl. *Internet of Things*, IoT) sehr eindrucksvoll. Dabei muss der Fokus auf dem Schaffen von Mehrwert für den Forscher liegen. Die Digitalisierung des Labors soll sich so vollziehen, dass sie dem Menschen ungeliebte oder fehleranfällige Aufgaben abnimmt und ihn in seiner Arbeit kontextabhängig unterstützt, ohne „im Weg zu sein“.

1.2 ZIELSETZUNG

Abbildung 1 zeigt die Arbeit im Labor nach heutigem Standard im direkten Vergleich zum digitalisierten Prozess, der das Ziel der vorliegenden Arbeit darstellt.

Es ist links der übliche Ablauf zu sehen. Das Protokoll wird vom Labormanager papierbasiert erstellt und dem Mitarbeiter übergeben, der seine Resultate händisch und auf Papier dokumentieren soll. Dabei muss er seine Arbeitsmaterialien immer wieder aus der Hand legen, um im Protokoll zu lesen, Berechnungen durchzuführen oder zu dokumentieren. Nach Ende des Ablaufs müssen die handschriftlichen Notizen in den Computer übertragen werden. Dabei wird kein Standard-Datenformat verwendet, so dass die Daten nicht vergleichbar oder direkt maschinell weiterverarbeitbar sind.

Rechts hingegen ist der Arbeitsablauf gezeigt, wie er im digitalisierten Labor möglich sein soll. Der Manager formuliert das Protokoll digital, sodass es vom Mitarbeiter im Labor direkt aufgerufen und durchgeführt werden kann. Während des Prozesses findet eine kontextbasierte Unterstützung des Arbeitenden statt. So können relevante Sicherheitsinformationen und kurze Hinweise zum aktuellen Arbeitsschritt beispielsweise direkt in einer Datenbrille angezeigt werden. Der Mitarbeiter hat beide Hände zum Arbeiten frei und kann mit dem Laborsystem z. B. über Sprachbefehle interagieren. Alle Schritte werden automatisch während der Durchführung dokumentiert, sodass das führen

handschriftlicher Laborbücher entfällt. Diese Dokumentation ist sofort digital verfügbar und findet in einem standardisierten Datenformat statt, das von weiteren Computerprogrammen zur automatischen Analyse gelesen werden kann. Das Labor kann ebenfalls Aufgaben, wie Ansatz- oder Kalibrationsberechnungen, übernehmen. Dadurch werden Fehler vermieden und eine gleichbleibend hohe Datenqualität sichergestellt. Da die gesamte Dokumentation live während des Prozesses stattfindet, entbindet das digitale System den Mitarbeiter von der manuellen Aufbereitung seiner Daten nach Abschluss der Arbeit. Stattdessen kann er sich anderen Aufgaben zuwenden, die in den meisten Fällen als angenehmer empfunden werden dürften.

Zur Realisierung dieses Ziels soll eine zuverlässige und belastbare IT-Infrastruktur konzeptioniert werden, die die Anbindung von Laborgeräten und anderen Ressourcen im Sinne einer IoT-Architektur erlauben. Diese Architektur soll flexibel und gut skalierbar sein, um eine unkomplizierte Anpassung an andere Laboratorien oder Arbeitsabläufe und Experimente zu erlauben. Dazu soll die Kommunikation zwischen den einzelnen Komponenten möglichst nach einheitlichen Standards stattfinden.

Zur Interaktion des digitalisierten Labors mit dem Nutzer sollen *Wearables* (Datenbrillen, Smartphones, Tablets, etc.) eingesetzt werden, da diese eine gute Unterstützung des Experimentators erlauben, ohne ihn in der Ausführung der Arbeiten zu behindern. Um eine Flexibilität in der Formulierung der Abläufe zu gewährleisten, sollen die Ablaufprotokolle für das digitalisierte Labor möglichst wenige direkte Abhängigkeiten zu den einzelnen Nutzerinteraktionsmedien aufweisen. Ebenfalls soll die Anbindung der Laborgeräte so weit abstrahiert werden, dass Implementierungsdetails der Geräteansteuerung keine Rolle spielen. So soll in dem digitalisierten Labor ein Gerät eines Herstellers ohne Änderungen in den Ablaufprotokollen gegen ein vergleichbares eines anderen Herstellers austauschbar sein.

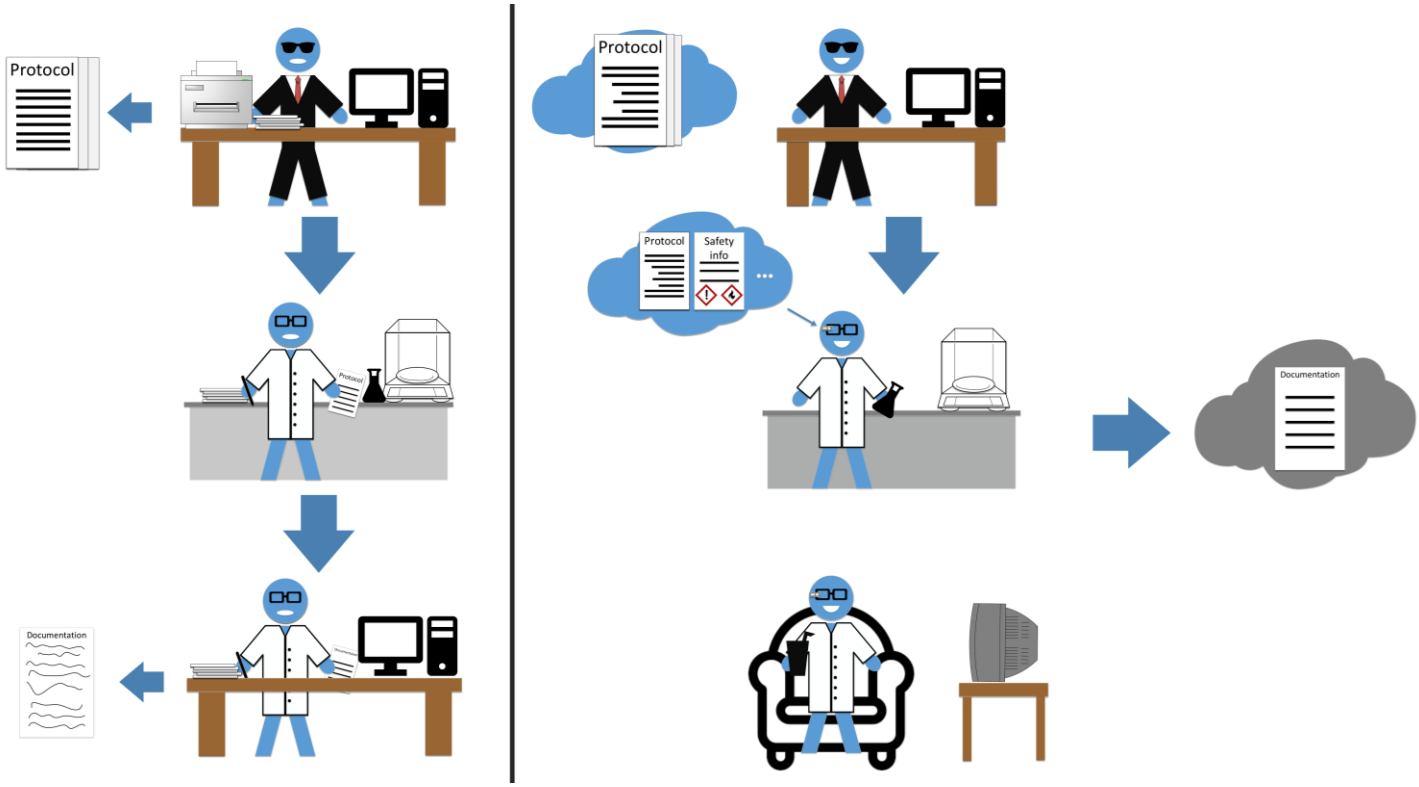


Abbildung 1: Schematische Darstellung der Laborarbeit zur Verdeutlichung des Ziels der vorliegenden Arbeit.

2 THEORETISCHE GRUNDLAGEN

Um die digitale Transformation des biotechnologischen Labors auf solider Basis durchführen zu können, sind verschiedene Technologien und Werkzeuge notwendig. Kapitel 2.1 (Seite 3) geht auf die Technologien zum Aufbau von digitalen Infrastrukturen ein und beschreibt auch die notwendigen Überlegungen bei der Kombination dieser Elemente zu funktionalen Einheiten. Darauf aufbauend beleuchtet Kapitel 2.2 (Seite 11) aktuelle Entwicklungen zur Digitalisierung im Laborumfeld sowie in diesem Kontext verschiedene Initiativen zur Standardisierung und deren technische Umsetzung.

2.1 IT-INFRASTRUKTUR

Die Systemarchitektur für komplexe Anwendungen, wie die digitale Transformation eines Labors, muss notwendigerweise auf leistungsfähigen Technologien der Informationstechnik aufbauen. Im Folgenden werden diese Technologien und ihre Implikationen auf verschiedene Aspekte der Systemauslegung beschrieben und vor dem Hintergrund ihrer Anwendung im digitalisierten Labor gezeigt.

2.1.1 NETZWERKE

Eine Verbindung zwischen mehreren Computern zum Zweck des Datenaustauschs wird als Netzwerk bezeichnet [40]. Die Verbindung kann dabei sowohl „netzartig“ in Form mehrerer Knoten, als auch sternförmig mit einem zentralen „Vermittler“ bestehen. Ein sternförmiges Netzwerk bietet eine größere Flexibilität, da die einzelnen Teilnehmer jederzeit getrennt und verbunden werden können ohne andere zu beeinflussen. Dabei werden die Teilnehmer durch wenige zentrale Knoten, sogenannte *Router*, *Switches* oder *Hubs,* angebunden [13]. Da es sich bei Mobiltelefonen, tragbare Endgeräten und *Embedded Computern* ebenfalls um Computer handelt, erstrecken sich Netzwerke heutzutage deutlich weiter, als dies noch vor einigen Jahren der Fall war [49]. Ein lokales Netzwerk (engl. *Local Area Network*, LAN) unterscheidet sich dabei vom globalen Internet nur durch seine Größe und physikalische Zugänglichkeit, nicht aber durch die eingesetzte Technologie [40].

Ein Teilnetz hat einen separaten Adressraum, der jeden Teilnehmer eindeutig identifiziert. Die Verbindung zwischen Netzen kann durch Vermittler, so genannte *Router*, hergestellt werden. Router sind Teil von mehreren Netzwerken und können so Daten zwischen den Netzteilen vermitteln [13].

2.1.1.1 BASISTECHNOLOGIEN

Ein wichtiges Merkmal der modernen Netzwerktechnik ist die paketvermittelte Datenübertragung. Im Gegensatz zu einer festen Punkt-zu-Punkt-Verbindung, wie sie früher z. B. beim Telefonieren durch das „Fräulein vom Amt“ hergestellt wurde, wird bei einer paketvermittelten Übertragung keine direkte oder exklusive Verbindung zwischen Sender und Empfänger benötigt. Diese haben feste Adressen und die Daten werden in Abschnitte („Pakete“) aufgeteilt, die jeweils Informationen über den Empfänger und ihren Kontext im Gesamtdatenstrom enthalten. Ist der Empfänger für den Sender direkt zu erreichen (im selben Teilnetz), so kann das Paket direkt zugestellt werden. Wenn ein Paket an einen Empfänger in einem anderen Netz adressiert ist, kann ein passender Router das Paket entsprechend weiterleiten. Ressourcen und Dienste werden in einem Netzwerk häufig von speziell für diesen Zweck ausgestatteten Computer, so genannten *Servern,* bereitgestellt [40].

Physikalisch basieren fast alle heute verbreiteten Netzwerke auf dem sternförmigen Ethernet-Verbindungsstandard nach IEEE (Institute of Electrical and Electronics Engineers) 802.3 oder 802.30 mit RJ-45 (Registered Jack 45) Kabeln. Immer weitere Verbreitung finden kabellose Funknetzwerke nach dem IEEE 802.11 Standard. Diese werden als WLAN (Wireless LAN) oder Wi-Fi bezeichnet. Router (in diesem Fall sog. *WLAN Access Points*) bilden die Brücke zwischen Funk- und Ethernet-Netzwerken [40].

2.1.1.2 DATENÜBERTRAGUNG

Zur Strukturierung der Datenübertragung wird neben der notwendigen physikalischen Verbindung eine standardisierte Möglichkeit der Datenübertragung benötigt. Hier findet nahezu ausschließlich der TCP/IP (Transmission Control Protocol/Internet Protocol) Protokollstapel (engl. *Protocol Stack*) Anwendung. Dieser ist in Abbildung 2A vereinfacht dargestellt. Abbildung 2B stellt die Ebenen des Stacks mit verschiedenen möglichen eingesetzten Protokollen dar und zeigt auch die unterschiedlichen historisch relevanten Hardware-Basistechnologien [36].

Die unterste Ebene ist die Netzwerkzugriffsebene. Sie wird durch die verwendete Hardware bestimmt. Dies ist z. B. CSMA/CD (Carrier Sense Multiple Access/Collision Detection) bei Ethernet [36].

Das *Internet Protocol* (IP) definiert die Adressierung und Paketierung der Daten. Die darüber liegende Tarnsportschicht sorgt für die korrekte Zustellung der Datenpakete vom Quell- zum Ziel-System. Hier kommen je nach Anwendungsgebiet das *Transmission Control Protocol* (TCP) sowie das *User Datagram Protocol* (UDP) zum Einsatz. TCP verfügt im Gegensatz zu UDP über Mechanismen zur Sicherstellung einer vollständigen Übertragung, ist dadurch aber auch bedeutend langsamer [36].

Die Anwendungsebene schließlich nimmt die übertragenen Daten entgegen und interpretiert sie je nach Anwendungsfall [36]. Die ältesten und meistverwendeten Protokolle dieser Ebene dienen dem Senden und Empfangen von Emails. Die erste Email wurde bereits 1972 durch Ray Tomlinson, einen Mitarbeiter eines Ingenieurbüros in Kalifornien, verschickt. Die Entwicklung des Internets wie es heute verstanden wird (in Form des *World Wide Web,* www) nahm hingegen erst 1989 am Europäischen Forschungsinstitut für Kernphysik (CERN) in der Schweiz durch den Briten Tim Berners-Lee seinen Anfang [40]. Inzwischen existieren unzählige Protokolle der Anwendungsebene, die jeweils speziellen Aufgaben dienen. Dabei müssen diese nicht monolithisch sein, sondern können wiederum selbst auf anderen Anwendungsebenen-Protokollen aufbauen [36].

2.1.1.3 KOMMUNIKATIONSPROTOKOLLE DER ANWENDUNGSEBENE

Einige der im Rahmen dieser Arbeit verwendeten oder angesprochenen Protokolle sollen in diesem Abschnitt genauer erläutert werden. Das *Hypertext Transfer Protocol (HTTP)* bildet als im Internet weit verbreitetes Protokoll zur Übertragung strukturierter Daten für die Anzeige von Webseiten auch die Basis vieler anderer höherer Protokolle. So werden *Representational State Transfer (REST)* und auch SOAP (früher: *Simple Object Access Protocol*) oft auf Basis von HTTP implementiert.

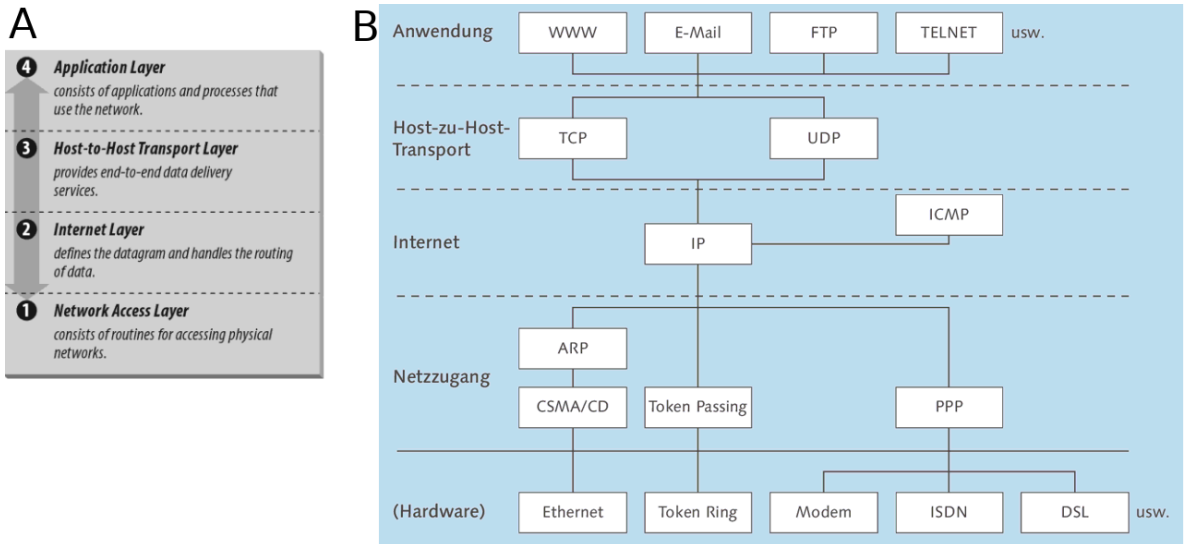


ABBILDUNG 2: A:VEREINFACHTE SCHEMATISCHE DARSTELLUNG DES TCP/IP PROTOCOL-STACKS. (BILDQUELLE: [36]) B: DETAILLIERTE DARSTELLUNG DES TCP/IP PROTOCOL STACKS MIT ANGABE DER GÄNGIGSTEN VERWENDETEN PROTOKOLLE UND BASISTECHNOLOGIEN. (BILDQUELLE: [40])

Eine besondere Rolle kommt den *Remote Procedure Call* (RPC) Protokollen zu, die nicht nur der Übertragung weiter verarbeitbarer Daten dienen, sondern zur direkten Steuerung von Programmen auf durch Netzwerke verbundenen Systemen dienen [52]. Programme werden so entwickelt, dass sie eine Vielzahl von Ablaufsträngen – je nach Art der Eingabedaten – wählen können. Dazu werden Einzelfunktionalitäten in sog. Funktionen oder Methoden strukturiert. Eine Funktion kann wiederum beliebig viele weitere Funktionen aufrufen. Diese müssen jedoch alle Teil desselben ausführbaren Programms sein [43]. Mit Hilfe von RPC Protokollen kann ein Server bestimmte seiner lokalen Funktionen über das Netzwerk aufrufbar machen. Ein Client kann diese Funktionen im Idealfall so verwenden, als ob es sich um lokale Funktionsaufrufe handeln würde [13].

Je nach Implementierung können unterschiedliche Übertragungsmethoden zur Realisierung eines RPC-Protokolls verwendet werden. Viele Implementierungen basieren auf der Übertragung von Datensegmenten in einer Auszeichnungssprache (engl. *Markup Language*), die einem bestimmten, festgelegten Muster folgen müssen. Beispielsweise verwendet JSON (*JavaScript Object Notation*)-RPC JSON-Nutzdaten (engl. *Payload*) [38], während XML (*Extensible Markup Language*)-RPC auf XML setzt [48]. Einen anderen Weg geht das neuere gRPC (Rekursives Akronym für *gRPC Remote Procedure Calls)*. Hier werden binäre Datenstreams erzeugt, die von Bibliotheksanbindungen so verarbeitet werden, dass für den Benutzer ein Verwendungsmechanismus entsteht, der von lokalen Funktionsaufrufen kaum mehr zu unterscheiden ist [32].

HYPERTEXT TRANSFER PROTOCOL

Das *Hypertext Transfer Protocol* wurde ursprünglich entwickelt, um die Übertragung von Webseiten in der Auszeichnungssprache *Hypertext Markup Language* (HTML) (vergl. Kapitel 2.1.1.4, Seite 7) zu ermöglichen. Damit bildet es auf Anwendungsebene das Basisprotokoll des www. Es handelt sich bei HTTP um ein Frage-Antwort-basierendes System, bei dem die vom Client (z. B. einem Webbrowser) gestellte Anfragen und die vom Server kommenden Antworten in einem bestimmten Format übermittelt werden [40]. HTTP wird (wie viele andere Internet-Standards) von der *Internet Society* (ISOC) in Form von RFCs (*Request for Comments*) veröffentlicht [27] und nach einem Review-Prozess in Form von *Internet Standards* definiert [53].

Sowohl Anfragen, wie auch Antworten besitzen einen Kopf (engl. *Header*), der in einem festgelegten Format die Parameter der Nachricht enthält. Darauf folgt der Nachrichtenrumpf (engl. *Body*),

```
GET /fachinfo/index.html HTTP/1.1
Accept: */*
Accept-Language: de, en-US
Accept-Encoding: gzip, deflate
User-Agent: Mozilla/5.0 (Macintosh; Intel Mac OS X 10_7_5)
AppleWebKit/536.26.17 (KHTML, like Gecko) Version/6.0.2 Safari/536.26.17
Host: buecher.lingoworld.de
Connection: Keep-Alive
```

```
HTTP/1.1 200 OK
Date: Tue, 16 Apr 2013 12:18:57 GMT
Server: Apache/2.2.9 (Unix)
Last-Modified: Sat 22 Sep 2012 10:21:37 GMT
ETag: "1b380f2-1ba9-454723b1"
Accept-Ranges: bytes
Content-Length: 7081
Content-Type: text/html

<!DOCTYPE HTML PUBLIC "-//W3C//DTD HTML
4.0//Transitional//EN">
<html>
<head>
[...]
```

ABBILDUNG 3: BEISPIEL EINER HTTP-KOMMUNIKATION ZWISCHEN CLIENT (WEBBROWSER) UND WEBSERVER. (BILDQUELLE: [40])

der die eigentlichen Daten beinhaltet. Anfragen können in HTTP mit verschiedenen Methoden (sog. *HTTP-Verbs*) eingeleitet werden, die vom Server unterschiedlich behandelt werden. So liefert eine GET-Anfrage die gewünschte Ressource aus. Eine POST-Anfrage sendet im Body zusätzlichen Payload. Dies können z. B. Eingabedaten eines Formulars auf einer Webseite sein. Abbildung 3 zeigt eine typische Anfrage und die darauf gesendete Antwort des Servers. Der Client fragt die Auslieferung einer Webseite an. Dabei gibt er weitere Parameter, wie den verwendeten Browser, Sprache und Encoding im Header an. Der Server sendet im Header einen Statuscode (in diesem Fall 200=OK), Informationen zum Server und Metadaten zum Inhalt. Als Payload folgt die eigentliche Webseite im HTML-Format. Dies ist das typische Vorgehen bei Webservern, die Webseiten an Browser ausliefern. Die vom Server gesendeten Inhalte können jedoch jedes beliebige Format haben, auch Binärdaten sind möglich [40].

HTTP liegt inzwischen in mehreren Versionen vor, die jeweils Ergänzungen oder Änderungen vornehmen, aber die Vorgängerversionen nicht ersetzen sollen. Die Version HTTP/2 nimmt einige Ergänzungen zur Erhöhung der Übertragungseffizienz vor und dient deshalb als Basis vieler höherer Anwendungsprotokolle [37].

REPRESENTATIONAL STATE TRANSFER

Bei der *Representational-state-transfer*-Technik handelt es sich um ein Software-Architekturkonzept zum Design von Webservices. Unter einem Webservice werden in diesem Zusammenhang Funktionen verstanden, die von einem Webserver zur Verfügung gestellt werden und primär dazu gedacht sind, von einem anderen Computerprogramm auf dem Client-Computer aufgerufen zu werden. So bietet z. B. Google einen Webservice an, um anderen Webseiten zu erlauben, Google-Dienste für ihr eigenes Angebot zu nutzen [40]. Das REST-Konzept wurde von Roy Thomas Fielding im Rahmen seiner Dissertation im Jahr 2000 entwickelt [26] und findet seitdem breite Anwendung bei Internet-Diensten [40].

REST-konforme (sog. RESTful) Webservices zeichnen sich durch eine uniforme Schnittstelle aus, mit der Clients vorab definierte statuslose (engl. *stateless*) Operationen auf Ressourcen durchführen können. Die möglichen Operationen müssen dem Client im Vorhinein bekannt sein und dürfen sich serverseitig nicht ändern. Zur Implementierung von RESTful Webservices wird oft HTTP eingesetzt, da es sich mit seinem Anfrage-Antwort-Modell an festen Adressen mit variablem Payload sehr gut eignet [26].

GRPC REMOTE PROCEDURE CALLS

Das gRPC-Protokoll ist ein in 2015 von Google ins Leben gerufenes Open-Source RPC Protokoll, welches auf Binärdatentransport über HTTP/2 basiert. Das Ziel von gRPC ist, entfernte Aufrufe möglichst genauso einfach zu gestalten, wie lokale. Dafür werden diverse Programmiersprachen direkt mit entsprechenden Bibliotheken unterstützt und Features, wie bidirektionale Datenstreams, Ablaufkontrolle und Authentifizierung angeboten [32].

```
service HelloService {
  rpc SayHello (HelloRequest) returns (HelloResponse);
}

message HelloRequest {
  string greeting = 1;
}

message HelloResponse {
  string reply = 1;
}
```

ABBILDUNG 4: BEISPIEL EINES IN *PROTOCOL BUFFERS* DEFINIERTEN GRPC-SERVICES. (BILDQUELLE: [32])

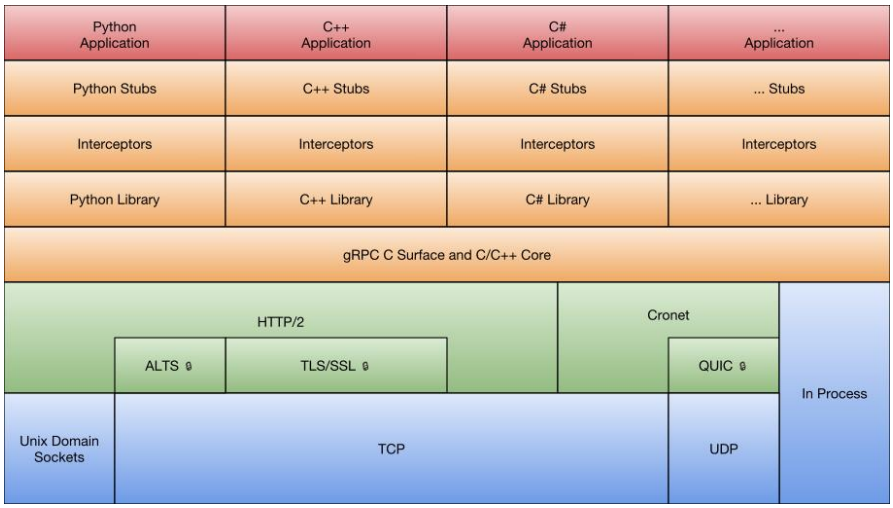


ABBILDUNG 5: DARSTELLUNG DES GRPC-PROTOKOLLSTAPELS UND -ARCHITEKTUR FÜR SPRACHEN, DIE AUF DIE GEMEINSAME BASISBIBLIOTHEK IN C/C++ ZURÜCK GREIFEN. (BILDQUELLE: [69])

Zur Definition von RPC-Schnittstellen bedient sich gRPC einer einfachen Beschreibungssprache, genannt *Protocol Buffers* [32, 63]. Abbildung 4 zeigt die Beschreibung eines einfachen RPC-Services. Die in *Protocol Buffers* geschriebenen „.proto-Dateien“ werden durch einen sog. „*Protobuf-Compiler“* zu Objekt- oder Funktionsdefinitionen einer Programmiersprache übersetzt. Es existieren *Protobuf-Compiler* für diverse Programmiersprachen, wie bspw. C++, Java, C# oder Python. Die so automatisch generierten Anbindungen in den jeweiligen Sprachen erlauben das Verwenden des RPC-Services mit einer Semantik, die dem lokalen Methodenaufruf in der jeweiligen Sprache gleicht. Die zugrundeliegende gRPC-Bibliothek übernimmt bei Verwendung dieser autogenerierten sogenannten „*Stubs*“ das Verpacken und Versenden der Nachrichten über das Netzwerk im Hintergrund. Mit diesem Mechanismus kann auch eine direkte Interaktion von Programmen, die in verschiedenen Sprachen entwickelt wurden, erreicht werden [32]. Da die Informationen als binärer Payload über eine HTTP/2-Verbindung geschickt werden, ist jedoch auch ein dynamisches (zur Laufzeit stattfindendes) Strukturieren dieser Daten möglich. Dabei kann auf die zur Übersetzungszeit statische Definition in *Protocol Buffers* verzichtet werden. Diesen Mechanismus macht sich bspw. die sila_tecan-Bibliothek zunutze [69].

Der Mechanismus der Datenübertragung von gRPC baut auf dem TCP/IP-Protokollstapel auf und verwendet für die Implementierungen in den verschiedenen Programmiersprachen teils unterschiedliche Architekturen. In Abbildung 5 ist beispielhaft der gRPC-Mechanismus für alle Sprachen gezeigt, die eine gRPC-Basisimplementierung in C/C++ als Ausgangspunkt verwenden. Die Anbindungsbibliotheken in den verschiedenen Sprachen rufen dabei die in der nativen gRPC-Core-Bibliothek enthaltenen Funktionen auf und bieten so den durch die *Protobuf Compiler* generierten *Stubs* diese Funktionen in der jeweiligen Sprache an. Die eigentliche Web-Anwendung kann diese *Stubs* dann je nach Bedarf verwenden. Die zugrundeliegenden Protokolle sind je nach Verwendungszweck austauschbar (z. B. TCP oder UDP) oder durch weitere Zwischenprotokolle ergänzbar (z. B. TLS (*Transport Layer Security*) bzw. SSL (*Secure Sockets Layer*) Verschlüsselung bei TCP-Kommunikation) [32].

2.1.1.4 AUSZEICHNUNGSSPRACHEN FÜR DIE DATENÜBERTRAGUNG

Für die Strukturierung von Daten, die z. B. über eine Netzwerkverbindung übertragen werden, haben sich verschiedene Auszeichnungssprachen (engl. *Markup Languages*) etabliert. Gemein ist all diesen Sprachen, dass sie textuelle Informationen in einer für Computerprogramme interpretierbaren Form strukturieren, die gleichzeitig für Menschen les- und schreibbar ist.

HYPERTEXT MARKUP LANGUAGE

Die *Hypertext Markup Language* wurde von Tim Berners-Lee 1989 zusammen mit dem dazu gehörigen Übertragungsprotokoll HTTP (siehe Kapitel 2.1.1.3, Seite 4) entwickelt. Es sollte ursprünglich die Strukturierung von wissenschaftlichen Informationen mit Querverweisen erleichtern. HTML und HTTP bildeten die Grundlage des www und werden bis heute für die Beschreibung und Auslieferung von Webseiten verwendet [40].

Abbildung 6 zeigt ein beispielhaftes HTML-Dokument – eine einfache Webseite. HTML definiert ein Element mit sog. „*Tags*“. Das „Wurzeltag“ `<html>` umschließt das gesamte Dokument. Das Ende eines durch einen Tag umfassten Bereiches wird durch den Namen des Tags mit vorangestelltem Schrägstrich gekennzeichnet. Ein HTML-Dokument zur Definition einer Webseite enthält einen `<head>` mit Metainformationen und einen `<body>`, der den eigentlichen, durch den Browser dargestellten, Inhalt spezifiziert. *Tags* können beliebig ineinander verschachtelt werden [68].

```
<!doctype html>
<html>
  <head>
    <meta charset="utf-8">
    <meta name="viewport" content="width=device-width, initial-scale=1.0">
    <title>Beschreibung der Seite (erscheint in der Titelzeile des Browsers)</title>
  </head>
  <body>
    <p>Dieser Text wird im Browserfenster angezeigt.</p>
  </body>
</html>
```

ABBILDUNG 6: BEISPIELHAFTES HTML-DOKUMENT, DAS EINE EINFACHE WEBSEITE BESCHREIBT. DER `<!DOCTYPE>`-*TAG* IST FÜR DAS BEZEICHNEN DES TYPS DER FOLGENDEN DATEN NOTWENDIG. (BILDQUELLE: [68])

JAVASCRIPT OBJECT NOTATION

Das Auszeichnungsformat *JavaScript Object Notation* wurde 2017 von Douglas Crockford in RFC 8259 spezifiziert und stellt ein text-basiertes Austauschformat für Objektdaten beliebiger Programmiersprachen dar [18]. Die Syntax lehnt sich an Konventionen aus objektorientierte Sprachen, wie JavaScript, C++ oder C# an, setzt solche jedoch nicht voraus [33].

JSON-Daten bestehen aus einer beliebigen Anzahl von Name-Wert-Paaren (engl. *Key-Value-Pairs*). Die Paare werden durch einen Doppelpunkt getrennt und mit Kommata aneinandergereiht. Der Name ist ein textueller Bezeichner, während der Wert verschiedene Datenformate annehmen kann. Neben Basistypen, wie Zahlen, Zeichenketten oder Wahr-Falsch-Werten, kann es sich dabei auch um verschachtelte JSON-Objekte oder Felder (engl. *Arrays*) handeln, die durch eckige Klammern ausgezeichnet werden. JSON-Objekte werden durch geschweifte Klammern umschlossen. Abbildung 7 zeigt ein JSON-Objekt, das ein Bild beschreibt. Durch die sparsame Verwendung syntaktischer Elemente sind JSON-Daten für den Menschen sehr gut lesbar [18].

```
{
  "Image": {
      "Width":  800,
      "Height": 600,
      "Title":  "View from 15th Floor",
      "Thumbnail": {
          "Url":    "http://www.example.com/image/481989943",
          "Height": 125,
          "Width":  100
      },
      "Animated" : false,
      "IDs": [116, 943, 234, 38793]
    }
}
```

ABBILDUNG 7: BEISPIELHAFTES JSON-DATENSET, DAS EIN BILD BESCHREIBT. (BILDQUELLE: [18])

Für die meisten modernen Programmiersprachen sind JSON (De-)Serialisierer verfügbar, die Objektdaten automatisiert in JSON-kodierte Zeichenketten umwandeln können oder die Felder einer (je nach Möglichkeiten der Sprache, zur Übersetzungszeit vorhandenen oder dynamisch zur Laufzeit erstellten) Objektstruktur mit den Werten aus einem JSON-Objekt füllen können [33]. Deshalb bietet JSON sich als Datenaustauschformat z. B. in Kombination mit REST-Webservices an [40].

EXTENSIBLE MARKUP LANGUAGE

Bei der *Extensible Markup Language* handelt es sich ebenfalls um eine textbasierte Auszeichnungssprache, die zum Datenaustausch zwischen Computerprogrammen entwickelt wurde. Sie wurde vom *World Wide Web Consortium* (W3C) 1998 veröffentlicht und ist nah mit HTML verwand. So verwendet sie ebenfalls *Tags* mit der gleichen Syntax wie HTML. XML selbst dient dabei als Metasprache zur Formulierung anwendungsspezifischer Auszeichnungssprachen [24]. Das als Weiterentwicklung von HTML entwickelte XHTML (*Extensible Hypertext Markup Language*) ist beispielsweise eine Neuformulierung von HTML in XML [40].

XML wird in ganz unterschiedlichen Zusammenhängen eingesetzt. So basieren verschiedene Dateiformate (z. B. SVG, *Scalable Vector Graphics* oder MPEG, *Moving Picture Experts Group*) zumindest teilweise auf XML. Abbildung 8 zeigt beispielhaft ein XML-Dokument, das vom Nationalen Wetterservice der USA (Vereinigte Staaten von Amerika), dem NOAA (*National Oceanic and Atmospheric Administration*), über eine Web-Schnittstelle abrufbar ist [68].

Domänenspezifische Auszeichnungssprachen werden in Form von *Document Type Definitions* (DTDs) oder Schemata (XSD, *XML Schema Definition*) verfasst. Der Verweis auf die verwendete DTD oder XSD kann im Kopf des XML-Dokuments angegeben werden, wodurch eine automatisierte Validierung des Inhalts gegen die Vorlage möglich wird. DTDs sind dabei die ältere Variante, die auf eine eigenen Syntax beruht, während XSDs als Nachfolgetechnologie moderner, flexibler und selbst in XML definiert sind [40].

```xml
<?xml version="1.0" encoding="UTF-8"?>
<current_observation>
[...]

<location>New York/John F. Kennedy Intl Airport, NY</location>
<station_id>KJFK</station_id>
<latitude>40.66</latitude>
<longitude>-73.78</longitude>
<observation_time_rfc822>Mon, 11 Feb 2008 06:51:00 -0500 EST</observation_time_rfc822>

<weather>A Few Clouds</weather>
<temp_c>-12</temp_c>
<relative_humidity>36</relative_humidity>
<wind_dir>West</wind_dir>
<wind_degrees>280</wind_degrees>
<wind_mph>18.4</wind_mph>

[...]

</current_observation>
```

ABBILDUNG 8: BEISPIELHAFTES XML-DOKUMENT ABGERUFEN VOM WETTER-SERVICE DES NOAA IN GEKÜRZTER FORM. (BILDQQUELLE: [68])

2.1.2 Virtualisierung

Virtualisierung beschreibt den Vorgang, Programme bzw. Betriebssysteme nicht direkt auf Computerhardware auszuführen, sondern von der Hardware einen Computer inklusive aller Komponenten simulieren zu lassen und auf dieser simulierten Instanz ein Betriebssystem und Anwendungssoftware auszuführen [61]. Auch wenn dieses Vorgehen zusätzliche Ressourcen zur Simulation benötigt, bietet es einige Vorteile. Die virtuellen Computer (engl. *Virtual Machines*, VMs) sind unabhängig von der realen Hardware und können problemlos auf andere Gastgeber-Systeme (engl. *Host Systems*) umgezogen werden. Außerdem lassen sich Gast-Systeme (engl. *Guest Systems*) beliebig während der Ausführung pausieren, kopieren und verschieben [19].

Um die virtuelle Umgebung zur Verfügung zu stellen, wird eine spezielle Software, ein sog. *Hypervisor*, benötigt. Dieser stellt auf Basis der physikalischen Systemressourcen des *Hosts* allen Gästen virtualisierte Umgebungen zur Verfügung, auf die die realen Systemressourcen dynamisch verteilt werden. Auf diese Weise können beliebig viele Gäste auf einem einzigen *Host* virtualisiert werden [19]. Moderne Hardware bietet Technologien, die es dem *Hypervisor* erlauben, mit möglichst wenig Geschwindigkeitsverlust diese Zuteilung durchzuführen [61].

Es kann sich bei dem *Hypervisor* sowohl um ein Programm handeln, das auf einem Betriebssystem neben anderen ausgeführt wird (vergl. Abbildung 9A) – z. B. bei der Virtualisierung auf einem normalen Desktop PC – , als auch um ein eigenes Betriebssystem, das ausschließlich dazu dient, eine virtuelle Umgebung für andere Systeme zur Verfügung zu stellen (vergl. Abbildung 9B) [19]. In letzterem Fall spricht man von einem „Virtualisierungsserver“. Die komplette Virtualisierung von Serversystemen hat den Vorteil, dass verschiedene Server-Anwendungen, die u. U. unterschiedliche Betriebssysteme benötigen, voneinander getrennt auf derselben Hardware ausgeführt werden können. Ferner werden die Systeme unabhängig von der konkreten Hardware, so dass sie im Falle eines Defekts ohne Probleme ihre Arbeit auf neuer oder redundanter Hardware wieder aufnehmen können [61].

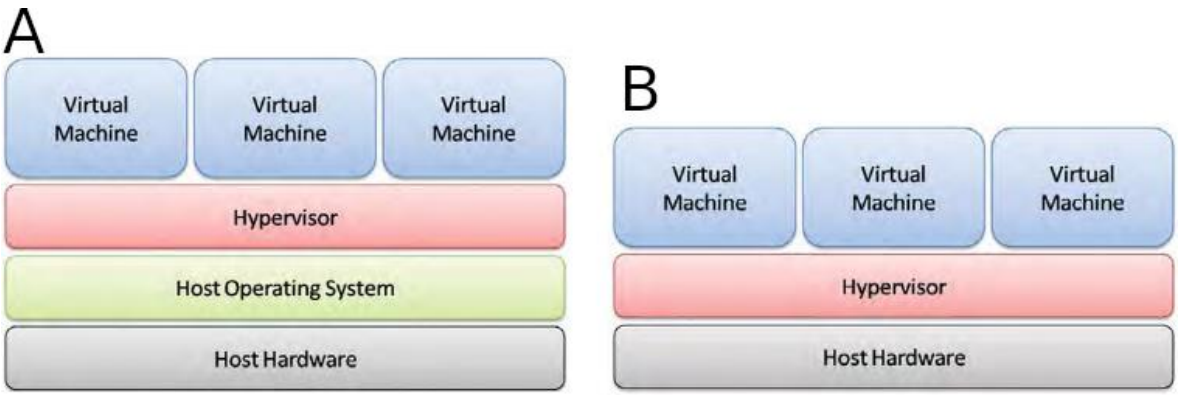


Abbildung 9: A: Der *Hypervisor* wird als normales Anwendungsprogramm auf einem Betriebssystem ausgeführt. B: Der *Hypervisor* ist Teil des Betriebssystems, das ausschließlich zur Bereitstellung von virtuellen Umgebungen für Gastsysteme dient. (Bildquellen: [19])

2.2 DIGITALISIERUNG UND DAS INTERNET OF THINGS

Seit Beginn der Computertechnologie in den 1950er Jahren hat die Rechenleistung kontinuierlich zugenommen, während die Größe der Rechenmaschinen im gleichen Zeitraum dramatisch sank. Seitdem ist ein immer weiterer exponentieller Anstieg („Regel von Moore“) zu beobachten (Abbildung 10). Durch die Limitierungen der Miniaturisierung in der Halbleitertechnologie durch Quanteneffekte, scheint ein Fortschreiten dieser Entwicklung nicht mehr möglich. Stattdessen konzentriert sich die Technologieindustrie seit einigen Jahren auf die immer bessere Integration der Computertechnologie in die physische Welt [42]. Durch diese Entwicklungen wurden das „Internet der Dinge“, als Synonym für das Eindringen der Computertechnologie in den Raum der physikalischen Gegenstände – und auch der Menschen – erst möglich.

Der rasante Fortschritt der Heimautomatisierung und die vierte industrielle Revolution haben die Konzepte des IoT und des *Industrial IoT* (IIoT) weit verbreitet und der technische Fortschritt geht im industriellen Bereich rasant voran. *Augmented Reality* Anwendungen helfen den Mitarbeitern bei der Kommissionierung [64] und Datenbrillen unterstützen Servicetechniker [35]. Die neuen Werkzeuge werden vor den Hintergründen des Arbeits- und Datenschutzes von staatlicher Seite evaluiert [14] und ihr Potential zur Steigerung der Produktivität untersucht [35]. Im Sinne von Mark Weiser (1991) ist ein fortschrittliches Computersystem eines, das in der Wahrnehmung des Benutzers in den Hintergrund tritt, da es die Erledigung von Aufgaben schnell und effizient ermöglicht, ohne dabei einen hohen Grad an Aufmerksamkeit zur Bedienung zu binden [78]. Diesen Anspruch erfüllen IoT-Anwendungen in vielerlei Hinsicht, da sie sich in den Arbeitsablauf flexibel integrieren und die Zeiten am stationären *Personal Computer* (PC) verringern.

2.2.1 DIGITALISIERUNG UND IOT IM LABOR

Auch wenn das Labor der Industrie in diesem Bereich um einige Jahre hinterherhinkt, so nimmt die digitale Transformation der biotechnologischen Laborwelt langsam Fahrt auf und kann – dort wo sie gewinnbringend implementiert wird – Wissenschaftlern und Arbeitern eine enorme Unterstützung sein [21, 25, 31, 34]. Aufgaben können durch das Wegfallen von manueller Konfiguration und Parametrisierung von Geräten schneller erfüllt werden [58]. Arbeiten werden durch besser strukturierte Arbeitsanweisungen und die Reduzierung von Schreibarbeiten einfacher [3]. Automatisierte Dokumentation erleichtert den Arbeitsablauf und reduziert mögliche Fehlerquellen [16, 30].

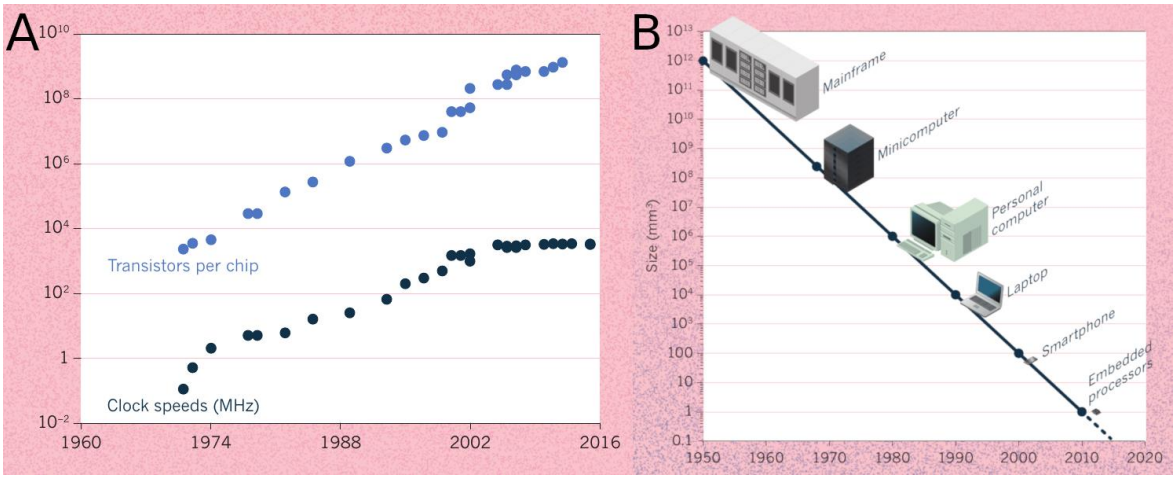


ABBILDUNG 10: ENTWICKLUNG DER COMPUTERTECHNOLOGIE. A: DIE TRANSISTORZAHL PRO CHIP HAT SICH ALLE ZWEI JAHRE UNGEFÄHR VERDOPPELT. EBENSO DIE TAKTFREQUENZ, BIS UNGEFÄHR 2004 AUFGRUND VON HITZEPROBLEMEN EIN FREQUENZMAXIMUM ERREICHT WAR. B: DA COMPUTERCHIPS SICH IN DER GRÖßE REDUZIERTEN UND GLEICHZEITIG AN RECHENLEISTUNG ZUNAHMEN, WURDE ETWA ALLE 10 JAHRE EINE NEUE FORM MÖGLICH. (BILDQUELLE: [42])

Labor-Informations- und Management-Systeme (LIMS) sind im hoch automatisierten Umfeld bereits weit verbreitet [67] und das Bewusstsein, dass automatisches digitales Datenmanagement beim Kampf gegen den Fachkräftemangel und für bessere Qualitätsstandards helfen kann, verbreitet sich immer weiter [30, 47, 67].

Steigende regulatorische Anforderungen und Initiativen, wie die FAIR (*Findable, Accessible, Interoperable, Reusable*)-Datenprinzipien [81, 82], benötigen schnelle und zuverlässige Methoden zur Akquise von Daten und dazugehörigen Metadaten (wie z. B. Geräteeigenschaften, Messhistorie, Umgebungsparameter, etc.) [67, 83]. Eine automatisierte Datenverwaltung eröffnet viele Möglichkeiten zur Zertifizierung – z. B. in Form GxP (*Good Manufacturing/Laboratory Practice*) [67]. Die produzierende Industrie hat diese Herausforderung bereits vor einigen Jahren angenommen und die digitale Transformation in Form der so genannten vierten Industriellen Revolution angestoßen [30]. Die Erfahrungen, die im Bereich des IIoT gesammelt wurden, helfen heute bei der Umsetzung der Digitalisierung im Laborbereich.

Laboratorien präsentieren sich jedoch im Gegensatz zu Fabriken in sehr unterschiedlichen Formen und müssen eine sehr viel flexiblere Planung und Änderung von Arbeitsabläufen unterstützen. Einerseits gibt es hoch automatisierte Labore mit begrenzter Mensch-Maschine-Interaktion [16], wie sie im Bereich z. B. der genetischen Forschung verbreitet sind. Hier werden viele Aufgaben durch Robotik-Plattformen durchgeführt, die gute Möglichkeiten zur Automatisierung und Anbindung von LIMS haben [29, 76]. Aber auch bei diesen Plattformen kann die Anbindung zusätzlicher Geräte oder Sensoren – sofern diese nicht direkt vom Hersteller der Plattform unterstützt werden – eine Herausforderung darstellen [30]. Viele Ansätze zur Labordigitalisierung konzentrieren sich stark auf solche hoch automatisierten Laborumgebungen [20, 47, 67].

Auf der anderen Seite stellen sich viele Labore eher in der Form einer „Werkzeugkiste für den Forscher“ dar. Viele (Klein-)Geräte, Materialen und Verbrauchsgüter sind in einem Raum verstreut und warten darauf, dass der Benutzer ihnen durch die Anwendung einer undefinierten Form von Standardarbeitsablauf (engl. *Standard Operating Procedure*, SOP) einen Sinn verleiht [29]. In diesem Umfeld ist eine gute horizontale Skalierbarkeit der Digitalisierungslösung notwendig, um schnell neue Arbeitsabläufe, Geräte oder Ressourcen einzubinden [7].

Eine besondere Herausforderung stellt die Kommunikation des digitalen Laborsystems mit dem Forscher oder Mitarbeiter dar [29]. Viele Ansätze zur Labordigitalisierung setzen momentan auf dezentralisierte Architekturen [3, 5], die diese Interaktionsebene nur begrenzt abbilden können. Besonders im biotechnologischen Labor, in dem komplexe Protokolle durchgeführt werden und viele zusätzliche Informationen durch den Forscher verarbeitet werden müssen, hat eine digitale Unterstützung das Potential, die Ergebnisdatenqualität zu erhöhen [47].

2.2.2 Standardisierte Kommunikation im Laborbereich

Zum Schaffen eines „Laboratory IoT“ müssen verschiedenste Systeme, Geräte und Services zusammenarbeiten. Peter Wegner (1996) beschreibt Interoperabilität von Softwaresystemen als „die Fähigkeit von zwei oder mehr Softwarekomponenten ungeachtet von Unterschieden in der Programmiersprache, den Interfaces oder der Ausführungsplattform zusammenzuarbeiten“; diese ist Voraussetzung für Flexibilität und gute Skalierbarkeit [77].

Um eine Interoperabilität von Laborkomponenten zu erreichen, ist die Verwendung eines einheitlichen Kommunikationsstandards notwendig [4]. Da sich ein solcher momentan noch nicht herausbilden konnte [58], müssen Lösungen zur zuverlässigen Kommunikation in diesem heterogenen Umfeld gefunden werden [60]. Hier können Gateway-Konzepte als Vermittlungsinstanz, wie sie sich bereits bei der digitalen Revolution in der Industrie bewährt haben [39], eine Möglichkeit sein.

Mit solchen Lösungen beschränkt sich die Verwendung heterogener und nicht standardisierter Protokolle auf den hardwarenahen Implementierungsbereich. Die Kommunikation der darüber liegenden Systeme verläuft standardisiert. Es haben sich mit SiLA2 (*Standardization in Lab Automation 2*) und OPC UA (*Open Platform Communications Unified Architecture*) – und darauf aufbauend LADS (*Laboratory Agnostic Device Standard*) – bisher zwei Ansätze für eine Vereinheitlichung der Laborgerätekommunikation gebildet. Ob sich einer – und wenn ja, welcher – dieser Standards weiter verbreiten wird, ist zum bisherigen Zeitpunkt nicht absehbar [30]. Im direkten Vergleich fallen jedoch viele Parallelen in beiden Standards auf, so dass sich eine Portierung von einmal standardisierten Anwendungen mit vertretbarem Aufwand realisieren lassen sollte. So teilen sich beide Konzepte in eine semantische Beschreibung und eine (austauschbare) technische Umsetzung auf.

2.2.2.1 STANDARDIZATION IN LAB AUTOMATION 2 (SILA2)

Der Kommunikationsstandard SiLA2 setzt sich zum Ziel, die Geräteansteuerung und die Datenstrukturierung im Labor zu vereinheitlichen und damit eine Kompatibilität zwischen den Geräten und Anwendungen verschiedener Hersteller zu schaffen [73]. Nach Erfahrungen mit dem Vorgängerstandard [4] wurde bei der Konzeptionierung besonderer Wert auf Flexibilität gelegt, um ein breites Anwendungsspektrum abbilden zu können. Die erste finale Version des Standards wurde im Oktober 2019 unter einer Open-Source-Lizenz (*Creative Commons Attribution-ShareAlike 4.0*, CC BY-SA 4.0) veröffentlicht [70–72].

SiLA2 wurde in drei unabhängigen Teilen verfasst (siehe Abbildung 11). Die Teile A [71] und B [72] bilden den stabilen Kern (*Core*), der seit der Veröffentlichung der Version 1.0 feststeht. Teil C [70] bildet eine sich ständig erweiternde Feature-Bibliothek. Teil A (*Overview, Concepts and Core Specification*) beschreibt die Konzepte zur Strukturierung der Datenströme ohne dies auf bestimmte technische Lösungen zu beziehen. Teil B (*Mapping Specification*) beschreibt die technische Umsetzung.

Konzeptionell bietet ein SiLA2 Server Gerätefunktionen im Netzwerk an und gibt SiLA2 Clients die Möglichkeit, Eigenschaftswerte abzurufen und Kommandos auszuführen. Funktionen und Eigenschaften von Geräten werden in so genannten *Features* zusammengefasst (Abbildung 12A). Ein Feature beschreibt die zusammenhängenden Attribute eines SiLA2 Servers (Abbildung 12B). Es enthält Geräteeigenschaften (*Properties*). Statische Eigenschaften verändern ihren Wert während der Ausführungszeit des Servers nicht, während dynamische Eigenschaften sich laufend ändern können.

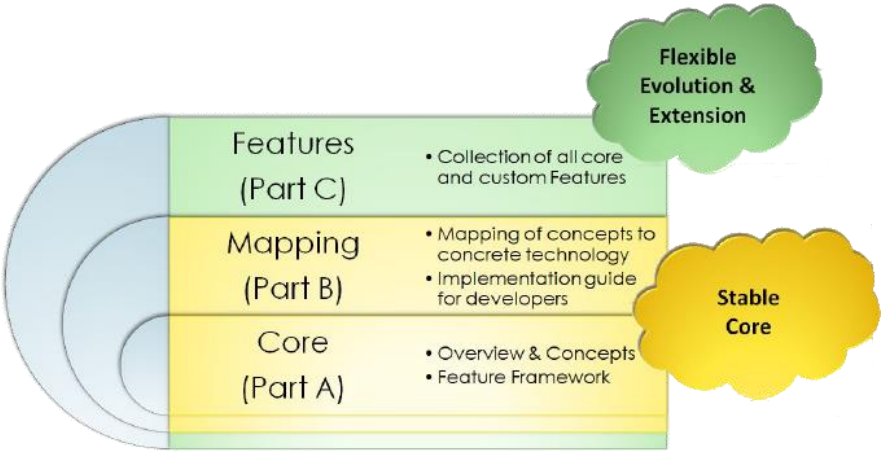


ABBILDUNG 11: TEILE DER SILA2 STANDARD-SPEZIFIKATION. TEIL A BESCHREIBT DIE KONZEPTE, TEIL B DEREN TECHNISCHE UMSETZUNG. ZUSAMMEN BILDEN SIE DEN STABILEN KERN. TEIL C IST EINE SICH STÄNDIG ERWEITERNDE FEATURE-BIBLIOTHEK. (BILDQUELLE: VERÄNDERT AUS [71])

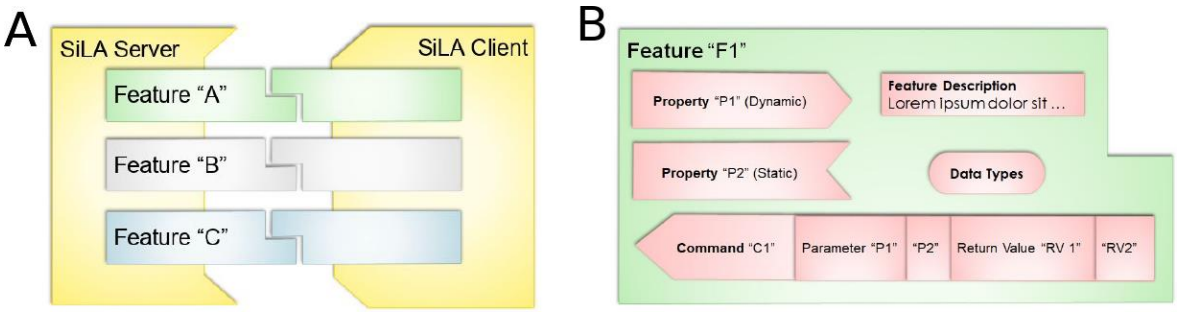


ABBILDUNG 12: SCHEMA DER SILA2 ORGANISATIONSEINHEITEN. A: EIN SILA2 SERVER BIETET BELIEBIGE FUNKTIONALITÄTEN, GRUPPIERT IN FEATURES, AN. EIN SILA2 CLIENT VERBINDET SICH MIT DEM SERVER UND FRAGT DIE VERWENDUNG DER FUNKTIONALITÄTEN AN. B: EIN FEATURE BESTEHT AUS EINER (FÜR DEN MENSCHEN LESBAREN) BESCHREIBUNG, BELIEBIG VIELEN KOMMANDOS UND EIGENSCHAFTEN SOWIE – WENN NOTWENDIG – SELBST DEFINIERTEN DATENTYPEN. (BILDQUELLE: [71])

Kommandos (*Commands*) werden vom Client gestartet und können eine beliebige Anzahl Parameter zur Spezifizierung akzeptieren. Kommandos übermitteln nach Ende ihrer Ausführung eine beliebige Anzahl Ergebnisse an den aufrufenden Client. Wenn lange Laufzeiten für Kommandos erwartet werden und/oder während der Ausführung Zwischenergebnisse dem Client übermittelt werden sollen, können diese als beobachtbar (*observable*) definiert werden. Der Client erhält dann während eines Kommandolaufs Statusinformationen (z. B. die Restlaufzeit) und Zwischenergebnisse vom Server. Ein Feature enthält außerdem eine detaillierte Dokumentation zu allen Funktionen in einer für den Anwender verständlichen Form [71].

Jeder SiLA2 konforme Server muss mindestens das SiLA2-Standardfeature anbieten und darf beliebig viele weitere Features implementieren. Das Standard-Feature bietet über einen standardisierten Satz von Kommandos und Eigenschaften jedem SiLA2 Client die Möglichkeit, die Dokumentation über alle vom Server angebotenen Features in maschinenlesbarer Form abzurufen [71].

Ein Magnetrührer würde neben dem obligatorischen Standard-Feature beispielsweise ein „Heizen"- und ein „Rühren"-Feature anbieten. Features dürfen sich dabei gegenseitig verändern – so kann ein „*Locking*"-Feature die Ausführung anderer Features nur für bestimmte, vorher angemeldete, Clients erlauben. Das „Rühren"-Feature hätte beispielsweise eine dynamische Eigenschaft „Drehzahl" und zwei Kommandos „Starten" (mit dem Parameter „Zieldrehzahl") und „Stoppen".

In der technischen Umsetzung basiert SiLA2 auf dem Open-Source *gRPC Remote Procedure Calls* Protokoll, welches plattformübergreifend zur Verfügung steht. Durch eine binäre, HTTP/2-basierte Übertragung ist gRPC schnell und universell einsetzbar. Implementierungen sind für die verschiedensten Programmiersprachen verfügbar [32]. SiLA2 beschreibt seine Datentypen sprachagnostisch und ermöglicht so die direkte Interoperabilität zwischen Software verschiedener Herkunft [72]. gRPC kann so angepasst werden, dass es auf Embedded-Prozessorarchitekturen einsetzbar ist [60], was die Einsatzmöglichkeiten von SiLA2 deutlich erweitert.

Neben dem SiLA2-Standard sind Referenzimplementierungen in den Sprachen Python, C++, C# und Java unter Open-Source-Lizenzen verfügbar [74]. Da der Standard offen zur Verfügung steht, steht jedem frei, eigene Implementierungen zu entwickeln. So existiert eine alternative C#-Implementierung (sila_tecan), die dynamisch den gRPC-Stream erzeugt und ohne Protobuf-Definitionsdateien auskommt [69].

2.2.2.2 OPEN PLATFORM COMMUNICATIONS UNIFIED ARCHITECTURE UND LABORATORY AGNOSTIC DEVICE STANDARD

Von der OPC (*Open Platform Communications*) Foundation wird ebenfalls ein Kommunikationsstandard für die Vernetzung von Geräten entwickelt. In der Industrie sind Datenaustauschschnittstellen

nach den klassischen OPC-Spezifikationen (*Data Access*: OPC DA, Alarms and Events: OPC AE, *Historical Data Access*: OPC HDA) weit verbreitet. Da diese Spezifikation jedoch auf Microsofts (*Distributed*) *Component Model* (COM/DCOM) basiert, ist sie auf Windows-Systeme beschränkt und im IoT-Umfeld nicht einsetzbar [23].

OPC UA (*Unified Architecture*) ist eine serviceorientierte und plattformunabhängige Weiterentwicklung dieses Standards auf Basis eines objekt-orientierten Konzepts. Dabei werden Maschinendaten im Gegensatz zum Vorgänger nicht nur transportiert, sondern auch strukturiert und in ihrer Semantik beschrieben. Als Basis dienen ein Metamodell zur Beschreibung der Basisstrukturen (Objekte, Datentypen, etc.) und eine Definition der Kommunikationsinfrastruktur auf technischer Ebene. Darauf basierend werden die konkreten Dienste und Anwendungen beschrieben (siehe Abbildung 13). Zur technischen Umsetzung werden nach aktuellem Stand ein auf TCP basierendes Binärprotokoll und ein SOAP (früher: *Simple Object Access Protocol*)-Webservice mit XML-Payload zur Verfügung gestellt [23].

OPC UA wurde in der Normenreihe IEC 62541 veröffentlicht. Anfangs standen die Spezifikationen und die Referenzimplementierung in ANSI-C [55] nur Mitgliedern der OPC Foundation zur Verfügung. Seit 2015 werden diese jedoch auch Nicht-Mitgliedern unter einer Open-Source-Lizenz zur Verfügung gestellt [54, 55]. Es existieren ebenfalls verschiedene (vielfach ebenfalls quelloffene) Implementierungen, unter anderem in C++ [2, 28], Java [56] und Python [28].

Die Zielrichtung der Formulierung von OPC UA liegt auf der Automatisierungstechnik und der Weiterentwicklung der industriellen Vorgängerstandards [23]. Dies schließt eine Anwendung im Laborbereich natürlich nicht aus, der Standard geht jedoch auf die spezifischen Anforderungen in diesem Bereich nicht ein. Der *Deutsche Branchenverband für Optik, Photonik, Analysen- und Medizintechnik* (SPECTARIS) hat sich in der Arbeitsgruppe „Vernetzte Laborgeräte“ die Schaffung eines standardisierten OPC UA Informationsmodells für die Labortechnik unter dem Namen *Laboratory Agnostic Standard* (LADS) zum Ziel gesetzt (vergl. Abbildung 14) [45, 46].

Nach aktuellem Zeitplan befindet sich diese Initiative momentan in der Phase des Verfassens der Spezifikationen und plant im Jahr 2022 mit der Entwicklung einer Referenzimplementierung zu starten. Ende 2022 soll diese komplettiert sein und LADS soll ab Ende 2023 für einen produktiven Einsatz im Labor zur Verfügung stehen [45].

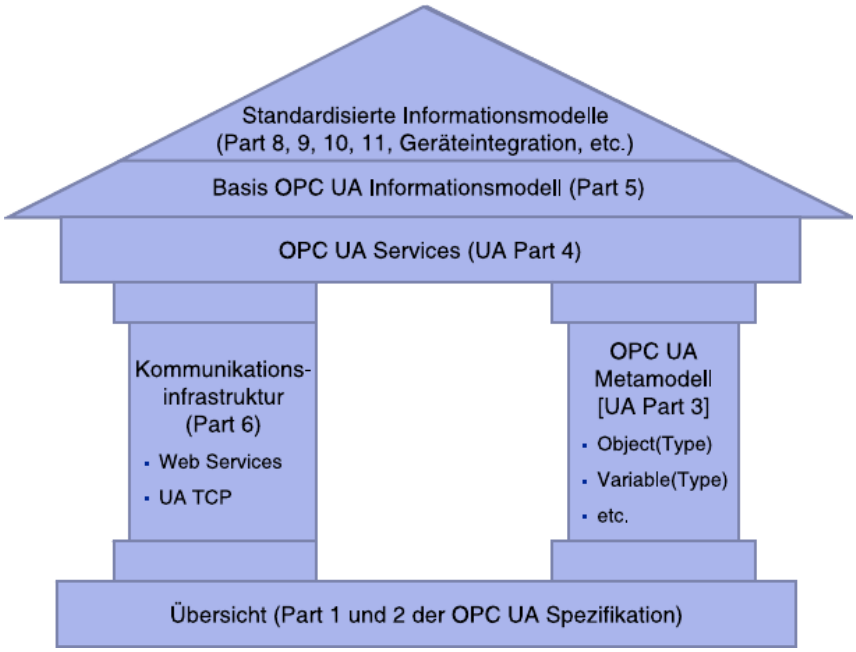


ABBILDUNG 13: AUFBAU DES OPC UA STANDARDS. (BILDQUELLE: [23])

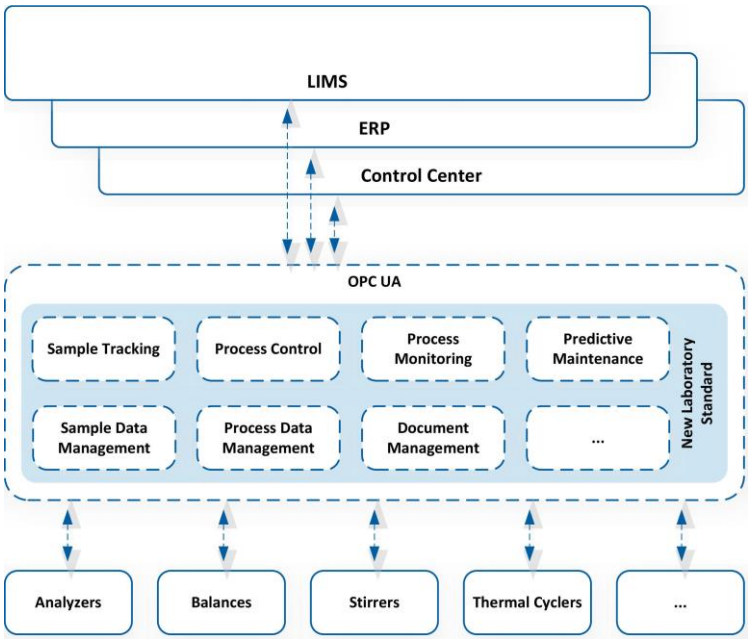


ABBILDUNG 14: UMFANG UND KONTEXT VON LADS ZUR STANDARDISIERUNG VON LABORGERÄTEKOMMUNIKATION. (BILDQUELLE: [46])

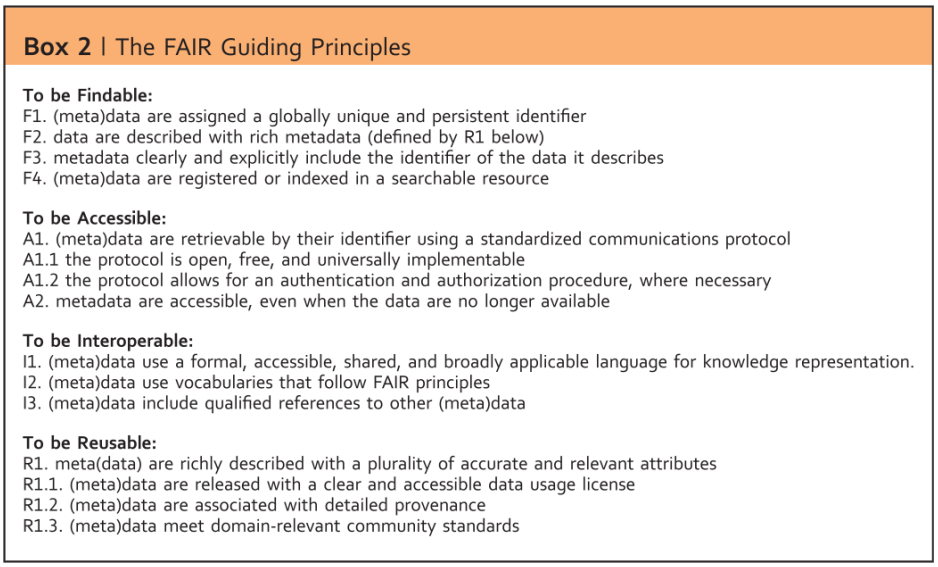
2.2.3 FAIR-DATA PRINZIPIEN

Die FAIR-Data-Initiative hat ihre Prinzipien zum zukunftsweisenden Umgang mit wissenschaftlichen Daten 2016 veröffentlicht [81, 82] und zwei Jahre später auf Basis der steigenden Akzeptanz weiter verfeinert [50]. Vier Kernforderungen werden formuliert: Daten sollen Auffindbar (*Findable*), zugänglich (*Accessible*), interoperabel (*Interoperable*) und wiederverwendbar (*Reusable*) sein. Dies schließt neben der Möglichkeit der Datenwiederverwendung durch menschliche Wissenschaftler ausdrücklich die maschinelle Verwendung durch Algorithmen, z. B. für Big-Data-Anwendungen, ein. Neben den Daten selbst kommt den dazugehörigen Metadaten eine große Bedeutung zu, da nur sie das Verständnis und die kontextuale Einordnung der Messdaten erlauben [82]. Eine Übersicht der Implikationen dieser Forderungen ist in Abbildung 15 gezeigt.

Um die Auffindbarkeit von Daten zu gewährleisten, muss einerseits sichergestellt sein, dass sie dauerhaft und zuverlässig zur Verfügung stehen. Dazu sind global eindeutige Identifikatoren und durchsuchbare Datenbanken notwendig. Gleichzeitig muss die „Suchbarkeit“ durch eine gute Beschreibung mit Metadaten gegeben sein, da ansonsten die Zugehörigkeit eines Datensatzes zur gesuchten Problemstellung eventuell nicht erkennbar ist [82].

Die Zugänglichkeit von Daten ist nach FAIR gegeben, wenn diese durch einen definierten Mechanismus mit entsprechender Berechtigung automatisiert erhalten werden können. Die dazugehörigen Metadaten sollten nach Möglichkeit ohne Berechtigung abrufbar sein, um das Suchen relevanter, geschützter Daten zu ermöglichen und auch sichtbar sein, wenn die Daten nicht mehr zur Verfügung stehen. Der Mechanismus zum Erhalt der Daten sollte mit einem standardisierten und offenen Kommunikationsprotokoll implementiert sein, um eine allgemeine Zugreifbarkeit sicherzustellen [82]. *„Accessible“* ist im Sinne von FAIR nicht mit „offen“ gleichzusetzen, da Zugriffsbeschränkungen, wo notwendig, ausdrücklich gestattet sind [50].

Interoperabel sind Daten nach FAIR dann, wenn sie in einer standardisierten und allgemein anwendbaren Form verfasst sind, die von Maschinen und Algorithmen interpretiert werden kann. Alle Verweise auf andere Daten müssen eindeutig und nachvollziehbar sein [82].



Box 2 | The FAIR Guiding Principles

To be Findable:
F1. (meta)data are assigned a globally unique and persistent identifier
F2. data are described with rich metadata (defined by R1 below)
F3. metadata clearly and explicitly include the identifier of the data it describes
F4. (meta)data are registered or indexed in a searchable resource

To be Accessible:
A1. (meta)data are retrievable by their identifier using a standardized communications protocol
A1.1 the protocol is open, free, and universally implementable
A1.2 the protocol allows for an authentication and authorization procedure, where necessary
A2. metadata are accessible, even when the data are no longer available

To be Interoperable:
I1. (meta)data use a formal, accessible, shared, and broadly applicable language for knowledge representation.
I2. (meta)data use vocabularies that follow FAIR principles
I3. (meta)data include qualified references to other (meta)data

To be Reusable:
R1. meta(data) are richly described with a plurality of accurate and relevant attributes
R1.1. (meta)data are released with a clear and accessible data usage license
R1.2. (meta)data are associated with detailed provenance
R1.3. (meta)data meet domain-relevant community standards

ABBILDUNG 15: ÜBERSICHT ÜBER DIE KERNFORDERUNGEN DER FAIR-DATA INITIATIVE. (BILDQUELLE: [82])

Um die Wiederverwendbarkeit wissenschaftlicher Daten zu erleichtern, sollten alle Daten und Metadaten akkurat beschrieben sein, nach allgemein in der jeweiligen Disziplin anerkannten Standards aufgenommen worden sein und die Herkunft sowie Lizenzfragen klar ersichtlich sein [82].

Viele wissenschaftliche Journale und staatliche Förderprogramme fordern das Einhalten der FAIR-Prinzipien inzwischen in unterschiedlichem Umfang [8]. So wird es von der DFG (Deutsche Forschungsgemeinschaft) für Projekte im Rahmen der Nationalen Forschungsdateninfrastruktur (NFDI) vorausgesetzt [51]. Auch die Europäische Kommission hat 2018 in Form eines Arbeitsplans die weiteren Schritte zur erfolgreichen Umsetzung der FAIR-Prinzipien veröffentlicht [17].

Um die für FAIR notwendige große Menge von Daten und vor allem assoziierten Metadaten ohne großen Arbeitsaufwand für den Forscher aufnehmen, verarbeiten, speichern und im Sinne von FAIR zur Verfügung stellen zu können, sind technische Lösungen zur Messgeräteanbindung dringend erforderlich [75]. Um die Daten interoperabel darstellen zu können, müssen Wege zur Disziplin-unabhängigen Beschreibung von wissenschaftlichen Daten gefunden werden. Momentan existieren zu diesem Zweck zwei verbreitete Ansätze. Die *Analytical Information Markup Language* (AnIML) beschreibt Daten in Textform und steht unter einer offenen Lizenz zur Verfügung [66]. Die *Allotrope-Foundation* hingegen formuliert ihren Standard auf Basis von Binärdaten und stellt die Spezifikationen nur Mitgliedern zur Verfügung [1].

2.2.3.1 ANALYTICAL INFORMATION MARKUP LANGUAGE

Die Analytical Information Markup Language ist eine standardisierte und maschinell lesbare Form der Daten-Strukturierung und -Speicherung für Messdaten. AniML wurde durch das ASTM (früher *American Society for Testing and Materials*) *Subcommittee E13.15 on Analytical Data* formuliert. Sie wurde ursprünglich für spektroskopische und chromatographische Daten entwickelt, ist aber flexibel gestaltet und universell einsetzbar [66].

AniML basiert auf textueller Datenspeicherung im XML-Format und organisiert sich in verschiedenen XML-Schemata (vergl. Abbildung 16). Ein *AniML Core Schema* definiert die Semantik für AnIML-konforme Dokumente und erlaubt eine automatisierte Validierung. AniML selbst definiert keine konkreten Strukturen für Messdaten, da diese sich je nach Anwendungsfall unterscheiden. Um AniML für eine spezifische Anwendung zu verwenden, wird ein *AniML Technique Definition Document* (ATDD) erstellt, welches die für diesen Fall benötigten Datenstrukturen definiert. Durch

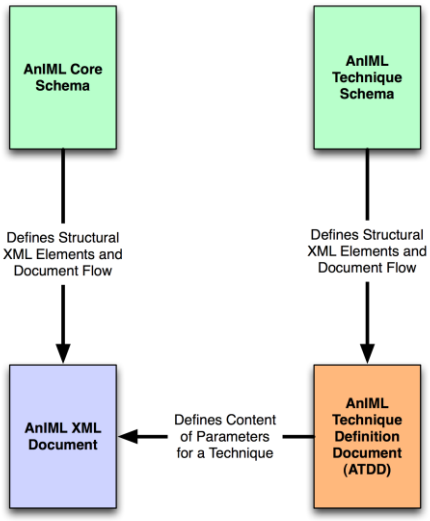


ABBILDUNG 16: ZUSAMMENHÄNGE DER DOKUMENTE EINER ANIML-DEFINITION. (BILDQUELLE: [79])

dieses Vorgehen ist AniML flexibel und universell einsetzbar. Das ATDD verwendet eine eigene Semantik, die wiederum im *AniML Technique Definition Schema* definiert ist. Technisch gesehen handelt es sich bei dem ATDD ebenfalls um ein Schema. Ein AniML-Dokument mit konkreten Messdaten eines in einem ATDD spezifizierten Versuchsablaufs ist sowohl gegen das *Core-Schema*, als auch gegen „seine“ *Technique Definition* validierbar [66].

Der Standard selbst umfasst dabei die vom ASTM unter einer quelloffenen Lizenz zur Verfügung gestellten Core- und Technique-Definition-Schemata [66]. Es existiert eine Bibliothek für ATDDs, die ständig erweitert wird. Momentan stehen z. B. ATDDs für Chromatographie-Anwendungen, UV/Vis (Ultraviolett/*Visible*, engl. sichtbar)-Spektroskopie und verschiedene chromatographische Detektorsysteme zur Verfügung [79]. Wenn Daten anderer Messverfahren mit AniML standardisiert strukturiert werden sollen, muss für diesen speziellen Anwendungsfall ein neues ATDD verfasst werden [66]. Durch den hohen Grad an Abstraktion und die Fokussierung auf maschinenlesbare XML-Schemata, lässt sich dieser Vorgang gut automatisieren. Da AnIML-Dokumente viele Metadaten und Informationen im sehr textreichen XML-Format enthalten, ist die Verwendung automatischer Generatoren, die die Messdaten in ein AniML-Konformes Format überführen, nahezu zwingend notwendig [65]. Um all diese Daten automatisiert während des Laborablaufs zu erhalten, müssen die verwendeten Geräte über entsprechende Mechanismen digital ansteuerbar sein [57].

Das von diesem Generator erstellte AniML-Dokument selbst dokumentiert die Ergebnisse eines spezifischen Messdurchgangs einer in der ATDD konkretisierten Methode. Es sind Informationen über die vermessenen Proben, wie zum Beispiel deren Herkunft oder Entstehungsweg, enthalten [66]. Damit lässt sich durch AniML leicht die Forderung der FAIR-Prinzipien nach einer umfassenden Annotation mit relevanten Metadaten erfüllen [82]. Außerdem sind die Messergebnisse selbst inklusive der Beschreibungen der zu ihrer Entstehung notwendigen Schritte enthalten. Wenn gewünscht, kann in einem AniML-Dokument ebenfalls ein *Audit Trail* der durchgeführten Änderungen und Digitale Signaturen aller Verwender gespeichert werden [66].

Abbildung 17 zeigt beispielhaft Ausschnitte der XML-Daten einer AniML-Strukturierung von Messdaten aus einer spektrometrischen Messung der Optischen Dichte bei 600 nm (OD_{600}).

Abbildung 17A stellt einen vereinfachten Ausschnitt des ATTDs zu dieser Methode dar. Neben den obligatorischen Angaben zum verwendeten Standard und Schema ist hier die Definition einer

A

```
<Technique name="OD600"
xmlns:xsi="http://www.w3.org/2001/XMLSchema-instance"
xsi:noNamespaceSchemaLocation="animl-technique.xsd"
version="0.90"extension="false">
<Documentation>Technique Definition for OD600, to estimate the
    concentration of bacteria at a wavelength of 600 nm</Documentation>
   <SampleRoleBlue-print name="Sample" samplePurpose="consumed"
    modality="required" maxOccurs="1" inheritable="true">
      <Documentation>The actual measurement sample</Documentation>
       <CategoryBlue-print name="Description" modality="required"
          maxOccurs="1">
           <Documentation>Parameters describing the
                experiment</Documentation>
           <ParameterBlue-print maxOccurs="1" name="Name"
                parameterType="String" modality="required">
               <Documentation>Name of the experiment</Documentation>
           </ParameterBlueprint>
       </CategoryBlueprint>
   </SampleRoleBlueprint>
</Technique>
```

B

```
<Result name="OD600 Absorbance">
   <SeriesSet name="Intensity" length="401">
       <Series name="Intensity" dependency="dependent"
          seriesID="Itensity_Page1" plotScale="none" seriesType="Float64">
           <IndividualValueSet>
               <F>0.3720</F>
           </IndividualValueSet>
           <Unit label="Absorbance">
               <SIUnit exponent="1" factor="1" offset="0">1</SIUnit>
           </Unit>
       </Series>
   </SeriesSet>
</Result>
```

ABBILDUNG 17: A: BEISPIELHAFTER AUSSCHNITT AUS DEM ATDD EINER OD_{600}-MESSUNG. B: BEISPIELHAFTE DARSTELLUNG EINES KONKRETEN OD_{600}-MESSWERTS. (DIE GRAFIKEN SIND VON FRAU MICHELLE ANGELICA DJUARI IM RAHMEN IHRER BACHELORARBEIT ANGEFERTIGT WORDEN.)

Messreihe mit einer OD_{600}-Messung gezeigt. Abbildung 17B zeigt einen Ausschnitt des AnIML-Dokuments zu einer konkreten Messung. Der optische Messwert selbst (0,3720 rel. AU) wird im Kontext seiner Entstehung verständlich eingeordnet, so dass diese Informationen für menschliche und maschinelle Verwender ohne weitere Zusatzkenntnisse verständlich und interpretierbar sind.

2.2.3.2 ALLOTROPE

Die Allotrope-Foundation hat sich zum Ziel gesetzt, einen Standard zu schaffen um experimentelle Daten in einem universellen Format zu strukturieren und zu verlinken. Das *Allotrope Data Format* (ADF) organisiert komplexe Daten in Form beliebig vieler mehrdimensionaler Arrays. Ziel ist, nicht nur die Ergebnisdaten selbst, sondern auch die zugehörigen Metadaten komplett abbilden zu können. Um Schemata für bestimmte Einsatzzwecke vorzugeben, sind so genannte *Allotrope-Ontologies* für konkrete Anwendungsbereiche, wie z B. Massenspektrometrie oder Chromatographie, verfügbar [1].

Das ADF basiert auf dem binären *Hierarchical Data Format 5* (HDF5). Dabei handelt es sich um ein frei und plattform-unabhängig verfügbares Datenformat, das speziell zur Speicherung großer Datenmengen entwickelt wurde [80]. Eine Allotrope-Datei im ADF-Format ist in drei Teile unterteilt (Abbildung 18). Die Datenbeschreibung (*Data Description*) enthält die Metadaten zur Einordnung und zum Verständnis der Messungen. Die Datenwürfel (*Data Cube*) enthalten alle Messdaten selbst in strukturierter und homogener Form. Das Datenpaket (*Data Package*) schließlich speichert die vom Messgerät direkt erzeugten Rohdaten [1].

Die Allotrope-Froundation stellt ihren Mitgliedern ebenfalls einen Datenexplorer sowie eine API (*Application Programming Interface*) zur Entwicklung für Software, die das ADF verwendet, zur Verfügung. Die Mitgliedschaft in der *Allotrope-Foundation* ist je nach Art der Institution kostenpflichtig und für einen Zugang zu den spezifischen Standardbeschreibungen sowie der API zwingend erforderlich [1].

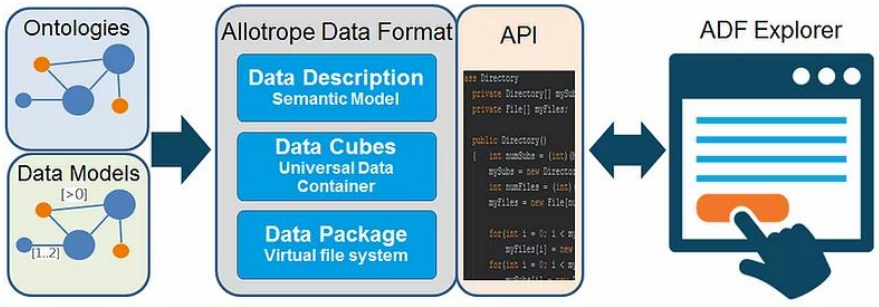


ABBILDUNG 18: AUFBAU DES ADF-DATEIFORMATS UND ALLOTROPE-ÖKOSYSTEM. (BILDQUELLE: [1])

3 PRAKTISCHER TEIL

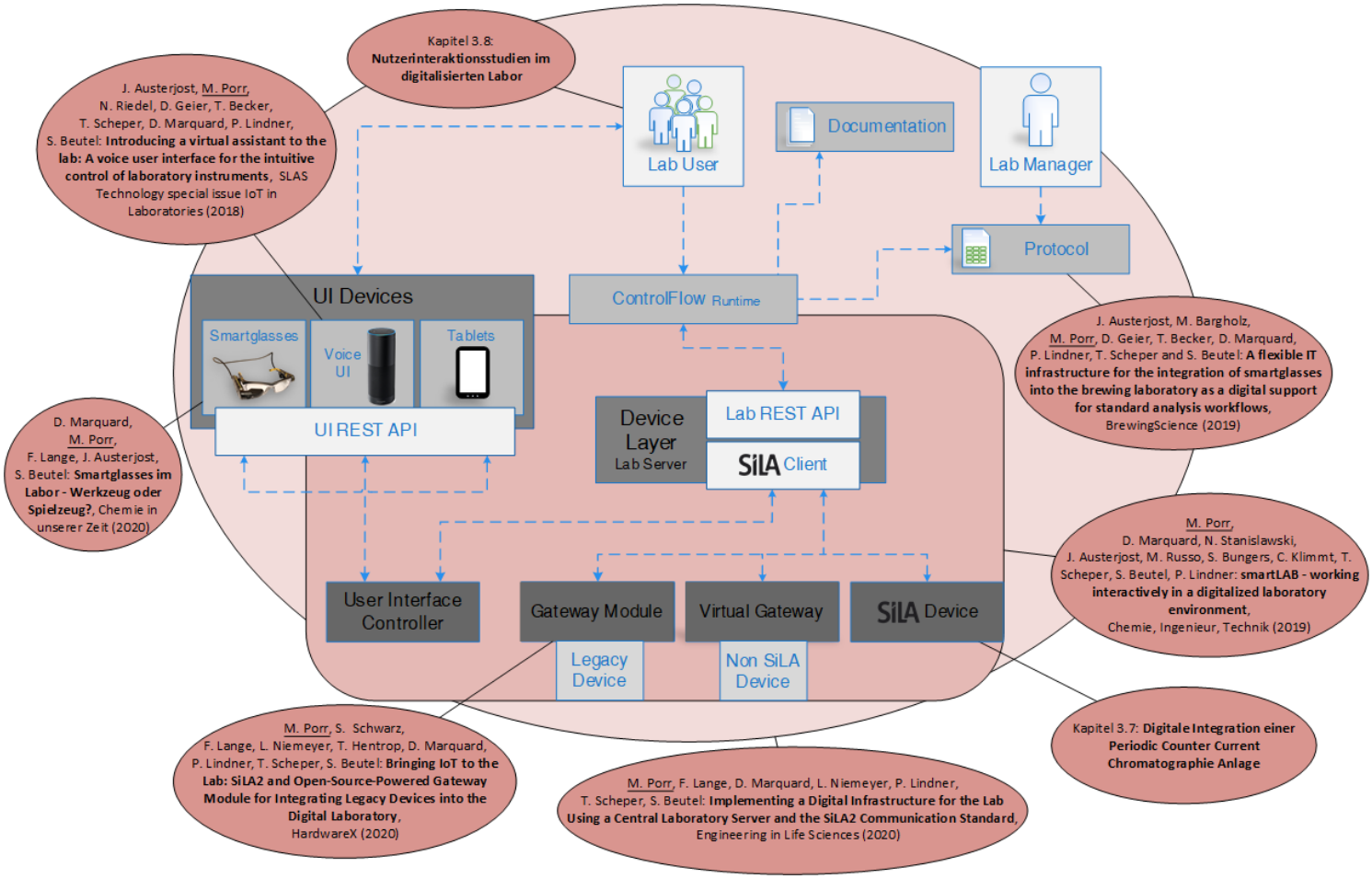


ABBILDUNG 19: ÜBERSICHT ÜBER DIE ENTWICKELTE DIGITALE INFRASTRUKTUR. DIE IN DEN EINZELNEN VERÖFFENTLICHUNGEN THEMATISIERTEN TEILBEREICHE SIND JEWEILS DURCH ENTSPRECHENDE ANMERKUNGEN (RUND) GEKENNZEICHNET.

Im Rahmen der vorliegenden Arbeit wurde ein Konzept zur umfassenden Digitalisierung im Laborbereich erarbeitet und praktisch umgesetzt. Abbildung 19 zeigt eine Übersicht über die entwickelte Infrastruktur und kennzeichnet die Teilbereiche, die in den im Folgenden aufgeführten Veröffentlichungen und Kapiteln detailliert thematisiert wurden.

Das digitalisierte Labor integriert die vorhandenen Geräte mit Hilfe eines einheitlichen Kommunikationsstandards (SiLA2). Geräte, die diesen Standard direkt unterstützen, können ohne weitere Maßnahmen angebunden werden. Geräte, die nicht SiLA2-fähig sind („*Non SiLA Devices*“ und „*Legacy Devices*“) werden mit Hilfe einer Vermittlungsinstanz (*Gateway*) angesprochen. Diese ist bei Geräten, die über eine Netzwerkschnittstelle verfügen, eine reine Softwareebene (*Virtual Gateway*), während bei Geräten, die nur über rudimentäre Schnittstellen verfügen (z. B. USB, *Universal Serial Bus* oder serielle Kommunikation) ein speziell zu diesem Zweck entwickeltes Hardware-Element (*Gateway Modul*) eingesetzt wird. Die Entwicklung und Verwendung dieses Gateway-Moduls sowie der Arbeitsablauf zur Anbindung von Laborgeräten mit SiLA2 wird in Kapitel 3.1 (Seite 22) beschrieben.

Ein zentraler Laborserver (*DeviceLayer*) bündelt die Datenströme und agiert sowohl als zentrale Anlaufstelle zum Labor, als auch als Datenbroker. Alle Geräte werden über das SiLA2 Protokoll angebunden und eine REST-API dient als Schnittstelle zum Labor. Geräte zur Nutzerinteraktion (UI-

Geräte), wie zum Beispiel ein *Head-Mounted-Display* an einer Laborschutzbrille („*SmartGlasses*“), Tablets oder Sprachsteuerungen werden abstrahiert über ein generisches Gateway (*User Interface Controller*) dargestellt. Kapitel 3.2 (Seite 54) beschreibt die grundsätzlichen Zusammenhänge dieser Infrastruktur sowie eine frühe Variante, die für die Messepräsentation „smartLAB III“ auf der Labor-Fachmesse Labvolution 2019 entwickelt wurde.

UI-Geräte werden über das generische Gateway angebunden, indem sie ihre Funktionalitäten abstrahiert und einheitlich in Form einer definierten REST-API anbieten, die vom UI-Controller verwendet wird. Die Verwendung von Sprachassistenten im Labor sowie eine beispielhafte Implementierung mit Hilfe eines *Echo*-Systems des Herstellers *Amazon Inc.* wird in Kapitel 3.1 (Seite 22) erläutert. Kapitel 3.4 (Seite 72) stellt die Möglichkeiten der Anwendung von SmartGlasses im Labor detailliert dar.

Zur Ausführung von Protokollen im digitalisierten Labor wird ein Prozessleitsystem (*Control Flow Runtime*) eingesetzt, das definierte Arbeitsablaufprotokolle liest und die entsprechenden Funktionen beim Laborserver über dessen REST-API anfordert. Diese Protokolle werden durch den Labormanager definiert und vom Forscher oder Labormitarbeiter ausgeführt. Die Kommunikation des Labors mit dem Forscher während des Prozessablaufs findet über die UI-Geräte statt. Eine Dokumentation aller Arbeiten kann voll automatisiert aus den jederzeit verfügbaren und standardisiert abgelegten Daten erfolgen. Kapitel 3.5 (Seite 85) beschäftigt sich mit der Standardisierung und Automatisierung von Laborabläufen und den notwendigen Voraussetzungen.

Kapitel 3.6 (Seite 95) beschreibt die Laborinfrastruktur in ihrer finalen Form und erklärt die Verwendung anhand eines anschaulichen Beispiels. Es wird basierend auf den vorangegangenen Arbeiten die Architektur des Gesamtsystems erläutert und die einzelnen Design-Entscheidungen vor dem Hintergrund der beschriebenen Ziele diskutiert. Die unterstützenden Informationen zu dieser Veröffentlichung (Kapitel 3.6.1, Seite 108) geben detaillierte Einblicke in die Funktion und Implementierung des Nutzerinteraktionssystems sowie des Prozesskontrollsystems.

Es wurde anhand einer am Institut entwickelten kontinuierlichen Chromatographieanlage ein Konzept für die digitale Integration eines bestehenden für Forschungszwecke entwickelten Systems gezeigt. Das Ziel dieser Arbeiten war vor allem, bereits bestehende Infrastrukturen, die im Laufe der Entwicklungszeit organisch gewachsen sind und sich bei komplexen Anwendungen oftmals sehr heterogen präsentieren, standardisiert digital ansteuerbar zu machen. Ebenfalls wurde anhand dieser Anlage die automatisierte Generierung von FAIR-konformen Ergebnisdaten durch das digitale Laborsystem etabliert. Gerade bei komplizierten Geräten, wie einem solchen Chromatographiesystem, ist eine leistungsfähige digitale Integration zur automatischen Aufnahme aller relevanten Daten und Metadaten nahezu unumgänglich, da sonst eine dem Benutzer nicht zumutbarer Dokumentationsaufwand entsteht. Diese Arbeiten sind in Kapitel 3.7 (Seite 118) beschrieben.

Zur Validierung der Einsatzfähigkeit und zur Verbesserung des Nutzererlebnisses wurde die Durchführung von Nutzerinteraktionsstudien wissenschaftlich vorbereitet und in ersten Tests bereits praktisch durchgeführt. Um die Akzeptanz der Benutzer für ein digitalisiertes Laborsystem zu steigern, sollte es nach Möglichkeit intuitiv bedienbar sein und den Verwender nicht in seiner Arbeit behindern. Die digitale Unterstützung sollte für den Anwender spürbare Vorteile – z. B. durch die Reduktion des Dokumentationsaufwands – haben. Entsprechend sollten Nutzerinteraktionsstudien vor allem die kontinuierliche Verbesserung des Systems in einer Art agilem Prozess zum Ziel haben. Die Arbeiten zu diesem Zweck sind in Kapitel 3.8 (Seite 125) beschrieben.

3.1 ENTWICKLUNG EINES GATEWAY-MODULS FÜR DIE ANBINDUNG VON BESTANDSGERÄTEN AN DAS DIGITALISIERTE LABOR

Zur skalierbaren und wartungsfreundlichen Anbindung von Geräten an das digitalisierte Labor ist die Verwendung eines einheitlichen Kommunikationsstandards entscheidend. Für die in dieser Arbeit vorgestellte Lösung wurde zu diesem Zweck der SiLA2 Kommunikationsstandard für Laborgeräte eingesetzt (siehe Abbildung 20).

Zum heutigen Zeitpunkt verfügen kommerziell erhältliche Geräte kaum über standardisierte Schnittstellen. Selbst Geräte des gleichen Herstellers weisen oft unterschiedliche Möglichkeiten zur digitalen Integration auf. Dieses Problem wird verstärkt durch die Notwendigkeit auch vorhandene Altgeräte und technisch einfache Geräte ohne leistungsfähige Schnittstellen ansprechen zu müssen. Um unter diesen Umständen ein schlankes Design des Laborservers und eine einfach horizontale Skalierbarkeit der Infrastruktur zu gewährleisten, wurde eine Microservice-basierte Architektur verwendet. Der Laborserver selbst kommuniziert ausschließlich mittels SiLA2 während die einzelnen Geräte über Gateways angebunden werden. Diese Gateways stellen die Funktionen des jeweiligen Geräts als SiLA2 Server zur Verfügung und kommunizieren mit dem Gerät selbst über das vorhandene heterogene Protokoll.

Für Geräte, die über eine Netzwerkschnittstelle verfügen, kann diese Gateway-Software als virtueller Container in dem zentralen Virtualisierungsserver ausgeführt werden. Bei Geräten, die nur über nicht-teilbare und lokal begrenzte Anschlussmöglichkeiten verfügen, kann diese Strategie nicht angewendet werden.

Um die digitale Integration solcher Bestandsgeräte und einfacher Geräte zu realisieren, wurde ein Gateway-Modul entwickelt, das basierend auf preiswerter Hardware die Netzwerkanbindung auch solcher Geräte ermöglicht. Das Modul ist für den Einsatz im Labor stabil und spritzwassergeschützt ausgelegt und basiert auf robuster Hardware, die den ununterbrochenen Einsatz ermöglicht. Das Design dieses Moduls wurde unter eine Open-Hardware-Lizenz gestellt und in dem im Folgenden abgedruckten Fachartikel veröffentlicht. Da SiLA2 selbst sowie das die technologische Basis bildende gRPC Protokoll unter einer Open-Source-Lizenz stehen, konnte auch sämtliche in diesem Rahmen entwickelte Software quelloffen zur freien Verfügung gestellt werden.

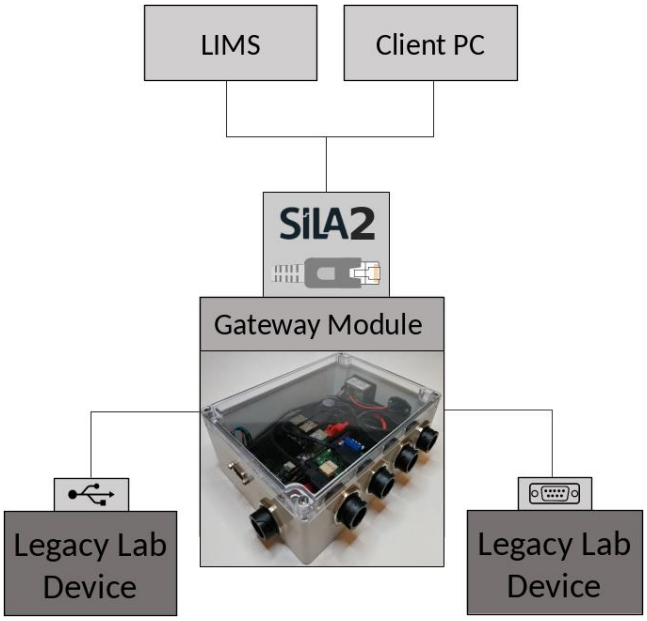


ABBILDUNG 20: GRAPHICAL ABSTRACT DES ARTIKELS "BRINGING IOT TO THE LAB: SILA2 AND OPEN-SOURCE-POWERED GATEWAY MODULE FOR INTEGRATING LEGACY DEVICES INTO THE DIGITAL LABORATORY".

Der folgende Artikel beschreibt außerdem detailliert das Vorgehen zur Etablierung und Implementierung einer Microservice-Architektur für SiLA2 Gateways. In diesem Rahmen ist auch ein System zum Verteilen und Steuern der entsprechenden Software-Bausteine vorgestellt, welches ebenfalls unter eine Open-Source-Lizenz veröffentlicht wurde. Zur Rationalisierung und einfachen Skalierung einer solchen Microservice-Architektur müssen die einzelnen SiLA2-Serveranwendungen einem einheitlichen Schema folgen und zentral gesteuert werden können. Zu diesem Zweck werden die einzelnen Microservices basierend auf einer gemeinsamen Vorlage erstellt und automatisiert auf die einzelnen Zielsysteme übertragen. Bei diesen Zielsystemen kann es sich sowohl um Gateway-Module, als auch um virtuelle Gateways handeln. Um eine zentrale Steuerung dieses Microservice-Schwarms zu gewährleisten, werden automatisiert Mechanismen zur Fernsteuerung der Services auf den Zielsystemen installiert. Über diese kann von einer zentralen Stelle aus die Steuerung, Wartung und Fehleranalyse für das gesamte Labor vorgenommen werden.

Eine ausführliche Dokumentation zum Bau und Betrieb des Moduls wird gegeben und durch die Verfügbarkeit von Skizzen, Plänen und Software als ergänzende Materialien [59] unterstützt. Sämtliche zu Bau, Konfiguration und Anwendung notwendigen Schritte sind ausführlich dokumentiert und ermöglichen auch anderen Anwendern eine Adaption des Konzepts für neue Einsatzbereiche.

Besonderen Wert wurde bei der Entwicklung des Moduls auf eine Eignung zum Einsatz auch unter realen Laborbedingungen gelegt. Die Elektronik wurde in einem entsprechend zertifizierten spritzwassergeschützten Gehäuse mit geeigneten Konnektoren untergebracht. Die Liste der verwendeten Komponenten und Zeichnungen der notwendigen Bearbeitungsschritte stehen ebenfalls zur freien Verfügung. Ferner wurde die unterbrechungslose Verwendung über mehrere Tage erfolgreich simuliert.

Zur Veranschaulichung der Verwendung des Gateway-Moduls wurden im Rahmen dieser Veröffentlichung Videos zur Erklärung aufgenommen und frei zur Verfügung gestellt. Mit Hilfe der QR-Codes in Abbildung 21 können diese Videos aufgerufen werden.

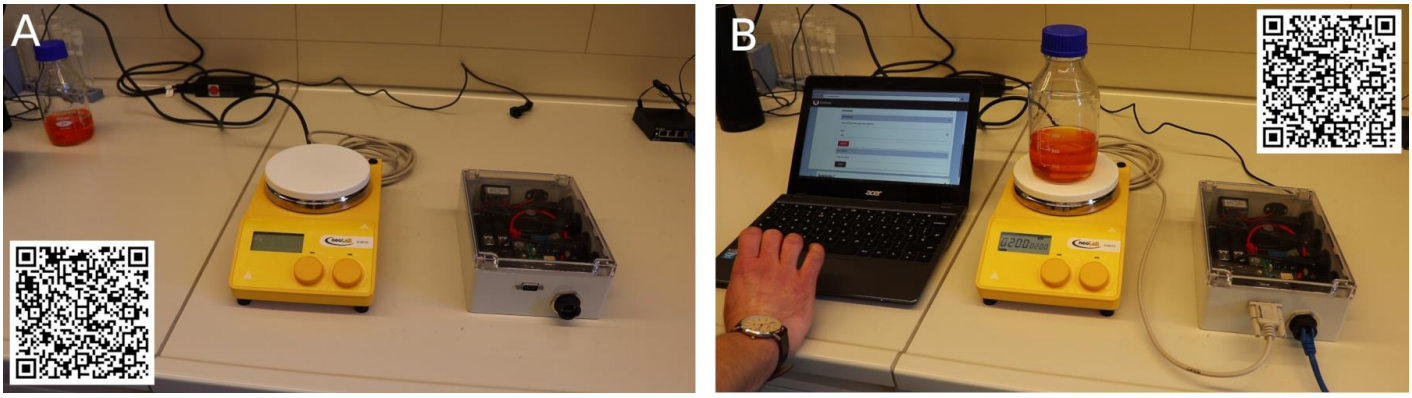


Abbildung 21: Zur Veranschaulichung des Aufbaus und der Benutzung des entwickelten Gateway-Moduls wurden Videos im Rahmen der Veröffentlichung öffentlich verfügbar gemacht. A: Erklärung des Aufbaus und Anschlusses eines Laborgeräts an das digitale Labornetzwerk mit dem Gateway-Modul. B: Verwendung des Gateway-Moduls und des daran angeschlossenen Laborgeräts. Das Gerät wird über SILA2 durch das Labornetzwerk gesteuert. Die Videos sind entweder nach Scannen der QR-Codes verfügbar, oder unter den folgenden Links.

A: https://data.mendeley.com/datasets/fvccdd6r4f/1/files/6eb9763f-b6e0-4a8d-a01b-7c916958a7f4/magneto_setup.mp4?dl=1;
B: https://data.mendeley.com/datasets/fvccdd6r4f/1/files/f125e65c-a726-4690-b7bb-e72aeec1b633/magneto_control.mp4?dl=1

HardwareX 8 (2020) e00118

Contents lists available at ScienceDirect

HardwareX

journal homepage: www.elsevier.com/locate/ohx

Bringing IoT to the Lab: SiLA2 and Open-Source-Powered Gateway Module for Integrating Legacy Devices into the Digital Laboratory

Marc Porr, Sebastian Schwarz, Ferdinand Lange, Laura Niemeyer, Thorleif Hentrop, Daniel Marquard, Patrick Lindner, Thomas Scheper, Sascha Beutel *

Leibniz University Hannover, Institute of Technical Chemistry, Callinstraße 5, 30167 Hannover, Germany

ARTICLE INFO

Article history:
Received 12 March 2020
Received in revised form 8 June 2020
Accepted 10 June 2020

Keywords:
Digital integration
Embedded computing
Network
Internet of things (IoT)
SiLA2
Laboratory
Digitization
Digitalization

ABSTRACT

In this article a gateway module to integrate legacy laboratory devices into the network of the digital laboratory in the 21st century is introduced. The device is based on ready to buy consumer hardware that is easy to get and inexpensive. Depending on the specific requirements of the desired application (bare embedded computer, RS232 serial port connector, IP65 certified casing and connectors) the needed investment ranges from about 95 € up to 200 €. The embedded computer runs an open source Linux operating system and can in principle be used to run any kind of software needed for communicating with the laboratory device. Here the open source SiLA2 standard is used for presenting the device's functions in the network. As an example the digital integration of a magnetic stirrer is shown and can be used as a template for other applications. A method for easy remote integration of the device to ensure an easy and consistent workflow in development, testing and usage is also presented. This incorporates a method for remote installation of SiLA2 servers on the box as well as a web frontend for administration, debugging and management of those.

Specifications table:

Hardware name	Gateway module for integrating legacy devices into the digital laboratory
Subject area	• Chemistry, Biochemistry and Biotechnology • Research Laboratories • Educational Tools and Open Source Alternatives to Existing Infrastructure • System Integration, Digitization and Digitalization

(*continued on next page*)

* Corresponding author.
E-mail address: beutel@iftc.uni-hannover.de (S. Beutel).

https://doi.org/10.1016/j.ohx.2020.e00118
2468-0672/

* (*continued*)

Hardware name	Gateway module for integrating legacy devices into the digital laboratory
Hardware type	• System Integration Gateway
Open source license	MIT-License: Copyright (c) 2020 Institut für Technische Chemie, Leibniz Universität Hannover
Cost of hardware	Minimal Version ≈ 95 € RS232 Version ≈ 110 € IP65 Version ≈ 200 €
Source file repository	Mendeley Data: "Source Files for tci-gatewaymodule Link: https://dx.doi.org/10.17632/fvccdd6r4f.1

1. Hardware in context

Today digitization and digitalization in chemical, biotechnological and biochemical laboratories is taking on drive and can – where implemented – help researchers and lab workers to work faster (less manual parametrization and configuration of devices), easier (better SOPs, less paperwork) and less error prone (automatic documentation) [1,2]. Laboratory information and management systems (LIMS) are common nowadays [3] and everybody in the field is aware that automated digital data management can help to overcome the lack of skilled labor and improve quality [4,2,3]. Increasing regulatory requirements and initiatives like the FAIR-data principles strongly rely on fast and dependable methods to acquire not only all process data available but also the corresponding metadata, like device properties and history of operations performed by a certain device or with a certain material [5,3].

In manufacturing industry the challenge was taken up a few years ago with the start of the so called fourth industrial revolution [2] and gateways similar to the one presented here have proven to pave the way to seamless device integration until all device manufacturers offer the needed interfaces and standards from the shelf [6].

1.1. Digitization in laboratories

Laboratories in contrast to factories present themselves in different fashions. On the one hand there are strongly automated ones that are common for example in genetic research. Here operations are highly automated and are usually carried out by robotic platforms that have good connectivity and automation protocols in place that enforce the implementation of LIMS infrastructures [7,8]. However, even in these labs the integration of hardware not directly supported by the robotics platform manufacturer can pose a challenge, like i.e. external temperature sensors or pumps [2]. The second variant of laboratories is more like a craftsman's toolbox. Several devices, tools and disposables are positioned in a room referred to as "laboratory" and wait for an operator to lend sense to them in a way that is described in a standard operating procedure [7].

How can devices, that were never made to be digitally integrated and are agnostic of each other and the context they are operated in, be used to automate workflows up to an extend where the operator can rely on data to be transferred and stored automatically?

1.2. A cost-efficient open-source solution

The industry has already taken up the challenge and many companies invest in new laboratory equipment that can be integrated into the network and thus enable a specially tailored LIMS to remote control the devices and acquire all measured data automatically [9]. In research and small laboratories, however, due to tight budgets, devices are kept in service for a longer time [10]. These legacy devices often can not be integrated into a network directly but usually offer some kind of low level communication method. Especially for scientist in universities these restrictions mean that they can not benefit from digital integration.

Presented here is a cost efficient gateway module (see Fig. 1) that can be used to integrate one or more legacy devices into the digital network and thus generate those benefits also for devices that were not originally designed to be used in a digitized lab.

In the presented solution, device integration is done using the emerging and open source SiLA2 standard [11]. A SiLA2-based method for easy remote integration of the gateway to ensure a flawless and consistent workflow in development, testing and usage is also shown. SiLA2 was chosen because of its benefits for scientists and developers. SiLA2 is an open source standard with several implementations in different programming languages available [12]. This availability is necessary for a use case like the one presented as heavy modifications in the code base were necessary. Furthermore, SiLA2 is specially designed for integration of laboratory devices and provides a logical base for device interaction in the laboratory.

However, the gateway module can be used to run any other kind of software as it is powered by an embedded Linux computer. For example IoT-technologies like MQTT can be also used as well as industry standards like OPC UA. In this case the hardware can be assembled the same way as it is presented here. The software has to be exchanged for tools which integrate with the standard of choice.

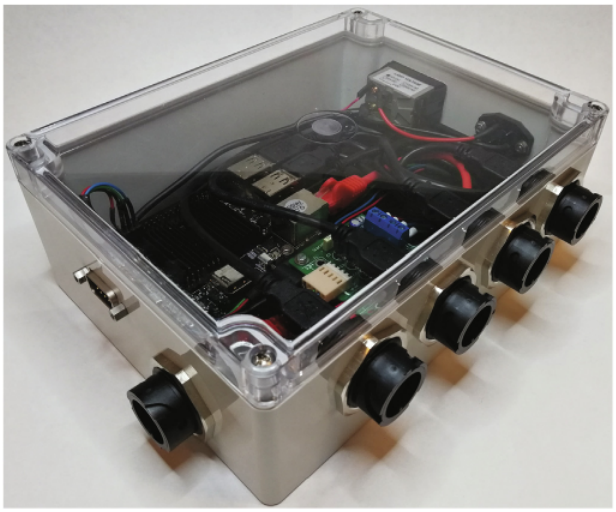


Fig. 1. The gateway module fully assembled in an IP65-certified splash-water proof casing.

2. Hardware description

Three different versions of the gateway module are presented that all run the same software but differ in the hardware used.

2.1. Minimal version

The most simple solution (referred to as "Minimal Version") only consists of a bare embedded computer (ODroid C2 single board computer from Hardkernel co., Ltd), a memory chip and a suitable power supply. This is enough to run the presented software and integrate almost any kind of legacy device. The overall cost of this solution was 93.02 € in December 2019.

The Minimal Version can also be used as a starting point for developing integrated devices that connect flawlessly into the lab infrastructure.

2.2. RS232 version

As for many legacy devices RS232-connections are common and USB-to-serial converters sometimes turned out to be unreliable, the "RS232 Version" features a RS232-standard serial connector normally used by labware manufacturers.

The total cost of this assembly was 108.71 € when parts were purchased in December 2019.

2.3. IP65 version

For applications where spillage of liquids may occur (as usually in chemical/biotechnological labs) or dust exposure is a problem, a version with a completely enclosed casing is presented. All components that are in contact with the box's exterior are certified as IP65 or higher according to IEC standard 60529. Thus this version will be referred to as the "IP65 Version".

It consists of a completely closed casing that protects the ODroid, its peripherals and the RS232 converter board. IP67 certified connectors are used to connect a power supply, the D-SUB RS232 terminal of the connector board, up to four USB2.0 ports and a RJ45 gigabit Ethernet connector. For easy operation of the encapsulated single board computer a power switch was integrated.

At a cost of 199.26 € (December 2019) even this highly endurable setup will often be a lot cheaper than buying a new device with the desired functionality. Buying device integration solutions from specialized firms will not only be much more expensive but also often leaves you with much less capable hardware that is hardly protected from harmful influences in the lab at all (also refer to Section 2.5).

2.4. Software

A ready-to-use system-image for the ODroid with all configurations and software presented in place is provided, that can be used to start device integration right away. Furthermore all software is available in git-repositories to set up your own tailored solution.

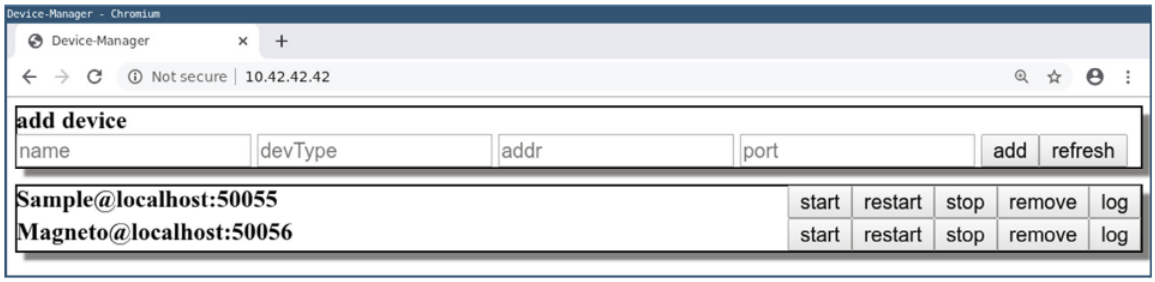


Fig. 2. Screenshot of the Device-Manager web frontend running on the gateway module. This can be used to conveniently manage, remote control and debug all SiLA2 servers running in the network.

As operating system for the ODroid the minimal Ubuntu 18.04.3 image from the official ODroid wiki [13] was used. When connected, the serial converter board can be used as */dev/ttyS1* without further configuration.

In contrast to some commercially available device integration solutions (see Section 2.5) the rapidly evolving open source SiLA2 standard for lab device communication is used for presenting the lab devices functions on the network. SiLA2 defines communication standards and data structures specific for laboratory devices on top of Google's open source remote procedure call protocol (gRPC) [11].

To enable easy SiLA2 server deployment, debugging and administration a system for remote publishing and remote control operations is also presented. A web frontend (see Fig. 2) for operating the gateway module over the network is preconfigured on the module itself when using the provided system image. This management system can also run on a dedicated server – which is especially useful when several gateway modules are in use in a large lab network.

2.4.1. Architectural overview

When using the proposed software architecture the gateway module is integrated into the laboratory network as described in Fig. 3. Legacy lab devices are connected to the gateway module which itself is integrated into the same network as the User PCs.

When developing a SiLA2 server the sila_tecan library is used. This has been extended by a mechanism to use a grpc-implementation that can run on the arm-architecture of the ODroid. The publish-system compiles the C# SiLA2 server-implementation on the User PC and publishes it to the gateway module. It also registers a systemd-service that is used

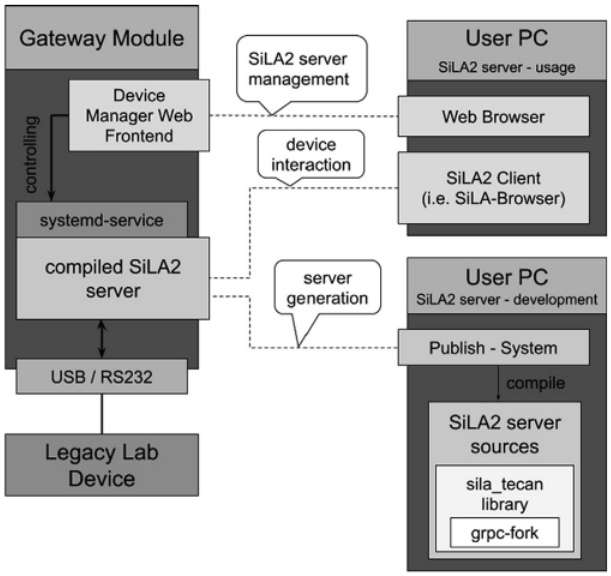


Fig. 3. Overview of the software architecture in the configuration presented. The physical location of all software components is highlighted. The Device-Manager web frontend is running directly on the gateway module in this example. During development the SiLA2 server gets published to the gateway module by the publish system. There it is controlled by anywhere in the network through the Device-Manager. All functionality of the device is now presented to the network via the SiLA2 server running on the gateway module. It can be utilized by any SiLA2 client anywhere on the network.

by the Device-Manager web frontend to control the SiLA2 server. This frontend can run on any device in the network, however, in the presented solution for reasons of simplicity it is running on the gateway module itself.

Once the SiLA2 server is installed on the gateway module this web frontend can be used from anywhere in the network to remote control all SiLA2 servers on the gateway module. To interact with the lab device a SiLA2 client is connected to the SILA2 server. In this work the sila-browser is used as a generic SiLA2 client to demonstrate the functionality of the gateway module.

2.5. Comparison to other solutions

Some companies offer similar devices. The Labforward GmbH (former Cubuslab/Labfolder) offers a combined hardware and service plan called "LabOperator" [14]. This includes an embedded computer that serves the same purpose as our gateway module and a server software for connecting these computers and planning/running protocols. Unfortunately this proprietary ecosystem forces the researcher to stick to a non-standard method of communication that is not open or extensible/modifiable.

In contrast the lab automation firm UniteLabs AG [15] offers embedded computing solutions for running open source SiLA2 servers named the "SiLA 2 Converter" [16]. Hardware-wise this converter is made of a home-use grade Raspberry Pi with all the drawbacks it brings along. UniteLabs offer these converters in combination with their service for device integration and driver development. Whereas such a service is attractive to companies, in research – especially in universities – funding for this is usually not available.

Furthermore none of these solutions are protected from dripping or sprayed liquids or offer significant dust shielding. This makes using these devices in labs unreliable and potentially dangerous.

2.6. Potential use to other researchers

The presented open source hard- and software may be useful to you when you ...

- ...are experimenting with digital interaction of legacy lab devices
- ...are looking for a cost effective method to start digitization and digitalization in your lab
- ...are working on i.e. SiLA2 integration of devices with an existing LIMS
- ...are implementing a LIMS or a digital control system with pre-existing hardware
- ...want to get insights into device integration with SiLA2
- ...do not want to pay for system integrator services just to be able to do your own integration/modifications anyway

3. Design files

Design filename	File type	Open source license	Location of the file
tci-gwm_cadmodel.iam	CAD-file	MIT	https://dx.doi.org/10.17632/fvccdd6r4f.1?urlappend=/files/26f4eda1-8059-4e06-ae35-9240d7dad593/tci-gwm_cadmodel.iam?dl=1
base_plate.ipt	CAD-file	MIT	https://dx.doi.org/10.17632/fvccdd6r4f.1?urlappend=/files/714998d3-e644-479a-b62f-1da49554d2ad/base_plate.ipt?dl=1
Bopla_M223G.ipt	CAD-file	MIT	https://dx.doi.org/10.17632/fvccdd6r4f.1?urlappend=/files/7585d432-5b41-40e3-9b65-a2e799573ca0/Bopla_M223G.ipt?dl=1
Bopla_M223G_cover.ipt	CAD-file	MIT	https://dx.doi.org/10.17632/fvccdd6r4f.1?urlappend=/files/e91f3ae4-f4d2-423f-a471-6ef39e881d46/Bopla_M223G cover.ipt?dl=1
M4x24x10.ipt	CAD-file	MIT	https://dx.doi.org/10.17632/fvccdd6r4f.1?urlappend=/files/68f3233f-2610-4a1d-ab48-0109889d268d/M4x24x10.ipt?dl=1
tci-gwm_wiring.pdf	drawing	MIT	https://dx.doi.org/10.17632/fvccdd6r4f.1?urlappend=/files/79170b6d-e581-44cc-99a0-431d19dd86c0/tci_gwm_wiring.pdf?dl=1

(*continued on next page*)

* (*continued*)

Design filename	File type	Open source license	Location of the file
tci-gwm_baseplate.pdf	drawing	MIT	https://dx.doi.org/10.17632/fvccdd6r4f.1?urlappend=/files/34d6bc-52ad-474e-88e1-c19dc582a6a8/tci-gwm_baseplate.pdf?dl=1
tci-gwm_holes.pdf	drawing	MIT	https://dx.doi.org/10.17632/fvccdd6r4f.1?urlappend=/files/36cbfd50-faaf-49fb-b665-629729c7277f/tci-gwm_holes.pdf?dl=1
tci-gwm_componentlist.xlsx	Spreadsheet	MIT	https://dx.doi.org/10.17632/fvccdd6r4f.1?urlappend=/files/a2c1628c-49bf-4340-b3d4-a86f5e31c24c/tci-gwm_componentlist.xlsx?dl=1
tci-gwm_ubuntu-18.04.3–3.16_odroid_systemimage_20200311.img.xz	system image	several OSS licences	https://dx.doi.org/10.17632/fvccdd6r4f.1?urlappend=/files/0b6e2f8e-87c9-4e80-b761-63eb1061ae7a/tci-gwm_ubuntu-18.04.3-3.16 odroid_systemimage 20200311.img.xz?dl=1
libgrpc_csharp_ext.aarch64.so	shared library	Apache 2.0	https://dx.doi.org/10.17632/fvccdd6r4f.1?urlappend=/files/df50d30c-b2-4643-b486-98a39ec5df71/libgrpc_csharp_ext.aarch64.so?dl=1
heavyload.csv	list	MIT	https://dx.doi.org/10.17632/fvccdd6r4f.1?urlappend=/files/df50d30c-b2-4643-b486-98a39ec5df71/libgrpc_csharp_ext.aarch64.so?dl=1
magneto_setup.mp4	video	MIT	https://dx.doi.org/10.17632/fvccdd6r4f.1?urlappend=/files/10ae6df5-0769-40af-a181-66b8c6c9f01e/heavyload.csv?dl=1
magneto_control.mp4	video	MIT	https://dx.doi.org/10.17632/fvccdd6r4f.1?urlappend=/files/6eb9763f-b6e0-4a8d-a01b-7c916958a7f4/magneto_setup.mp4?dl=1
gateway-publish.git/**/*	git-repository	MIT	https://dx.doi.org/10.17632/fvccdd6r4f.1?urlappend=/files/f125e65c-a726-4690-b7bb-e72aeec1b633/magneto_control.mp4?dl=1 (Repository-Link: https://gitlab.uni-hannover.de/tci-gateway-module/gateway-publish.git)
device-manager.git/**/*	git-repository	MIT	https://dx.doi.org/10.17632/fvccdd6r4f.1?urlappend=/files/37246900-bb3d-4e2f-ba42-604ce4970941/gateway-publish.zip?dl=1 (Repository-Link: https://gitlab.uni-hannover.de/tci-gateway-module/device-manager.git)
grpc.git/**/*	git-repository	Apache 2.0	https://dx.doi.org/10.17632/fvccdd6r4f.1?urlappend=/files/137e1a56-060d-4ce9-9996-8caabd44a491/device-manager.zip?dl=1 (Repository-Link: https://gitlab.uni-hannover.de/tci-gateway-module/grpc.git)
magnetosiladriver.git/**/*	git-repository	MIT	https://dx.doi.org/10.17632/fvccdd6r4f.1?urlappend=/files/7659c0f5-a869-45d1-b4f2-6f8e899381f2/grpc.zip?dl=1 (Repository-Link: https://gitlab.uni-hannover.de/tci-gateway-module/magnetosiladriver.git)
sila_tecan.git/**/*	git-repository	BSD 3-clause	https://dx.doi.org/10.17632/fvccdd6r4f.1?urlappend=/files/044d6140-a3f0-46c1-8152-28d3fec29306/sila tecan.zip?dl=1 (Repository-Link: https://gitlab.com/SiLA2/vendors/sila_tecan)

3.1. Design file descriptions

tci-gwm_cadmodel.iam Assembly file for the IP65-Version's casing containing of:

base_plate.ipt Model of the base plate from Pertinax hard paper plate

Bopla_M223G.ipt Model sowing the positions and sizes of holes machined into the Bopla M233 G base casing (modified from original model of M233 G [17])

Bopla_M223G_cover.ipt Model of Bopla M233 G casing's cover lid [17]

M4x24x10.ipt Model of M4x24x10 screws used for assembling Bopla M233 G [17]

tci-gwm_wiring.pdf Drawings with instructions on how to wire all electronic components for the three different versions of the gateway module

tci-gwm_baseplate.pdf Cutting template for the base plate from Pertinax hard paper plate

tci-gwm_holes.pdf Mechanical modifications template for the Bopla M233 G casing

tci-gwm_componentlist.xlsx Excel Spreadsheet with the full bill of materials including part numbers and distributor links for purchase

tci-gwm_ubuntu-18.04.3–3.16_odroid_systemimage_20200311.img.gz System image file for the ODroid C2 for a quick start with the gateway module (contains all modifications and software we present in this article)

libgrpc_csharp_ext.aarch64.so gRPC native library for C# compiled for aarch64 (ARM 64 Bit) architecture

heavyload.csv Temperature readings of the internal thermal sensor of the ODroid during heavy load test

magneto_setup.mp4 Video showing the setup steps to use the gateway module for controlling a magnetic stirrer

magneto_control.mp4 Video of the gateway module controlling a magnetic stirrer

3.2. Software repositories

The listed git repositories contain the software that was developed or modified for building the gateway module. The first four repositories are located in the *tci-gateway-module*-group on the gitlab-server of the *Leibniz University Hannover*: https://gitlab.uni-hannover.de/tci-gateway-module. The *sila_tecan* repository is located on *GitLab*: https://gitlab.com/SiLA2/vendors/sila_tecan. A "snapshot"-version of every repository at the time of submission can also be found in the supplementary data repository in the *Repository-Snapshots*-Folder. The following list also states the current commit hashes of the relevant branches' HEADs at the time of submission.

gateway-publish *[branch="master", HEAD = 0c7a57785bac143396f52188158c74fd252d3102]* Publish system for convenient SiLA2 server development for the gateway module

device-manager *[branch="master", HEAD = bce5059c8605c92b8afd68cc9c237694718e114b]* Web frontend for remote controlling and debugging SiLA2 servers running on gateway modules

grpc *[branch="arm-platform", HEAD = 0d417d55e95ef37cb7e42673b72687dc6e84f5eb]* Fork of the official gRPC repository (https://github.com/grpc/grpc) with modified library loading system for C# that allows using native libraries compiled for architectures different from x86 or x64 (in branch "arm-platform")

magnetosiladriver *[branch="master", HEAD = cce4efa7cd877c89c8eae256b7cbf9de8967bc2b]* SiLA2 server for controlling a *neoLab D-6010* magnetic stirrer

sila_tecan *[branch="tci-gwm", HEAD = 45b977b66d871b94beaea328c0771bdceff3d3c3]* Open source SiLA2 implementation in C# (used for all servers presented in this article) with modifications to exchange the gRPC implementation (in branch "tci-gwm")

4. Bill of materials

The following list is only a short summary. The complete bill of materials with manufacturer's component codes, links to online-shops and full component descriptions is to be found in the design files: a2c1628c-49bf-4340-b3d4-a86f5e31c24c/tci-gwm_componentlist.xlsx.

4.1. Minimal version

Designator	Component	Number	Cost per unit [€]	Total cost [€]	Source of materials	Material type
ODroid	ODroid C2 microcontroller	1	59.50	59.50	Hardkernel Co., Ltd.	electronic component
eMMC	16GB eMMC storage expansion	1	15.70	15.70	ALLNET GmbH	electronic component
DC Plug Power Assembly	DC power connector Plug Cable Assembly	1	1.12	1.12	Hardkernel Co., Ltd.	electronic component
Power Supply	GST40A05 5 V power supply	1	16.70	16.70	MEAN WELL Co., Ltd.	electronic component
			Overall cost:	93.02		

4.2. RS232 version

Designator	Component	Number	Cost per unit [€]	Total cost [€]	Source of materials	Material type
All components of Minimal Version				93.02		
D-SUB Connector	15–006543 D-SUB 9-pole connector	1	9.75	9.75	CONEC Elektronische Bauelemente GmbH	electronic component
MAX3232	MAX3232 CPE TTL/RS232 microchip	1	1.99	1.99	Maxim Integrated	electronic component
RS232 converter board	RS232/TTL converter board 810036	1	3.95	3.95	Pollin Electronic GmbH	electronic component
			Overall cost:	108.71		

4.3. IP65 version

Designator	Component	Number	Cost per unit [€]	Total cost [€]	Source of materials	Material type
All components of RS232 and Minimal Version				108.71		
Bopla M223 G	Enclosure	1	35.90	35.90	Bopla Gehäuse Systeme GmbH	ABS, PC
RJ45 Coupler	RJ45 inline Coupler 17–10019	1	7.95	7.95	CONEC GmbH	electronic component
0.25 m Ethernet Cable	0.25 m Ethernet Cable 855 W-003	1	0.40	0.40	ALCASA Elektronik AG	electronic component
USB Coupler	USB inline Coupler 17–200001	4	5.99	23.96	CONEC Elektronische Bauelemente GmbH	electronic component
0.5 m USB Cable	0.5 m USB Cable 2212-EU005	4	0.95	3.80	ALCASA Elektronik AG	electronic component
Belden Receptacle	Power Connector Receptacle G30A5M	1	1.80	1.80	Belden Inc.	electronic component

Belden Seal	Enclousure Power Connector Seal G30E-2	1	0.27	0.27	Belden Inc.	electronic component
Belden Connector	Enclousure Power Connector G30WF	1	2.35	2.35	Belden Inc.	electronic component
Rocker Switch	Rocker Switch SPST RND 210–00526	1	4.95	4.95	RND Components	electronic component
Pertinax Plate	Pertinax Hard Paper Plate 500x260 x1.5 mm	1	7.90	7.90	Masterplatex	PFCP201 (Hp 2061)
Spacer Tube	Spacer Tube d = 7 mm l = 5 mm	6	0.05	0.30	reichelt elektronik GmbH & Co. KG	PS
M2.5x12mm Screw	M2.5x12mm Screw	8				metal
M2.5 Nut	M2.5 Hexagon Nut	8				metal
M2.5 Shim	M2.5 Shim	8				metal
3.9x9.5 Metal Screw	3.9x9.5 Metal Screw	3				metal
M4.3 Shim	M4.3 Shim	3				metal
Estimated overall cost of assembly materials				0.97		metal
			Overall cost:	199.26		

5. Build instructions

5.1. Potential safety hazards

Please apply all necessary safety measures when soldering, wiring, cutting or drilling.

5.2. Embedded computer choice

The ODroid C2 single board computer from Hardkernel co., Ltd (see [18] for detailed hardware-specs) is used as it offers some advantages above other solutions. Compared to the omnipresent Raspberry Pi 3 B+ the ODroid offers a significant benefit in power [19]. The RaspberryPi 4 now has about the same specs but the ODroid makes using eMMC-storage-chips possible that have proved to be more reliable in 24/7 use than the Pi's SD-cards.

The ODroid C2 also is cheaper and has more processing power than the devices from the open BeagleBoard-series. But where the use of completely open hardware is of special importance, the ODroid can easily be swapped for a BeagleBone.

The ODroid C2 was chosen to run an open source SiLA2 implementation on a Linux system. If Windows software is necessary, change the ODroid C2 for a RaspberryPi 3 B+ with Windows IoT (Windows IoT is not compatible with the RaspberryPi 4).

5.3. Power supply assembly

The ODroid can be powered by a micro-USB cable plugged into the USB-OTG port. However, this is not recommended for 24/7 use – especially in an enclosed housing – as it increases heat production. Remove the jumper from J1 for power supply by the barrel connector [20].

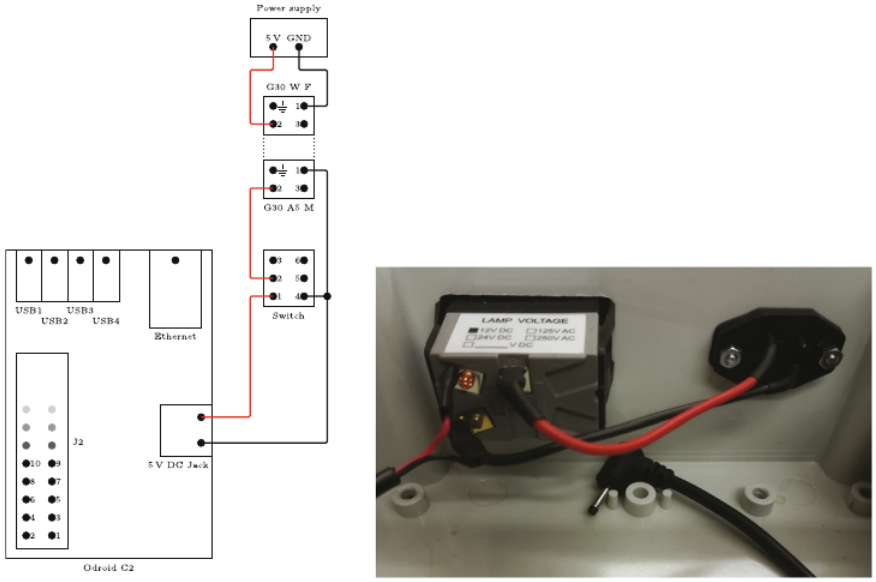
5.3.1. Minimal and RS232 version power supply assembly

For the Minimal and RS232 Version the 5 V side of the Power Supply is connected to the DC Plug Cable Assembly. Connecting this to the barrel connector on the ODroid board will power it up.

5.3.2. IP65 version power supply assembly

When setting up the Power Supply for the IP65 Version refer to Fig. 4 for graphical instructions and follow the Tasks as stated here.

When several gateways are to be run close together (i.e. in the same room) a combined power supply can be installed. That means connecting all the gateways in parallel to one suitable power source. When assuming 800 mA of power consumption in full load (see [21] for ODroid power consumption stats) and adding a good margin for transmission losses and external consumers (as USB devices) one of the 5 A Power Supplies used here should be able to power at least three boxes. Of course any other 5 V power supply can be used.



(a) Wiring diagram. (b) Photo of wired Switch and Receptacle.

Fig. 4. Graphical instructions for IP65 version's power supply assembly.

Task 1: mount the Belden receptacle and rocker switch to the casing. For power supply of the IP65 Version the IP67 certified Belden Receptacle has to be mounted inside the casing. Also a Rocker Switch in the 5 V wire was used for convenient power-on and off. This can of course be omitted of not necessary.

Task 2: solder the Belden receptacle. For the complete assembly the DC Plug Cable Assembly's +5 V wire (red) is soldered to the Rocker Switch's pin 1. The GND wire (black) is soldered to pin 4 of the Switch. Pin 4 of the Switch is connected with pin 1 of the male Belden Receptacle. Pin 2 of the Switch is connected with pin 2 of the Receptacle.

Task 3: solder the Belden connector with the power supply. Finally connect the Power Supply to the female Belden Connector. Solder pin 2 to +5 V (red) and pin 1 to GND (black).

5.3.3. Alternative IP65 version power supply assembly with cable gland

Alternatively – when no pluggable power connection is needed – the power hook-up can be done similar to the Minimal Version and an IP67 certified cable gland can be used to seal the casing.

5.3.4. Functional check

Before carrying out any other steps please make sure the ODroid works as expected. Simply connect the power supply. The red LED should light up and indicate that power supply is working.

5.4. Operating system installation

To test all other functions of the ODroid an operating system is needed. The ODroid has a standard eMMC connector that is used to connect a 16 GB eMMC chip. For a minimal (headless) operated Linux system the ODroid wiki states 4 GB as a minimum recommended disk size [13]. If you want to use the provided image a minimum of 6 GB is necessary. Alternatively a micro-SD-card can be used. But as eMMCs offer significantly improved performance over SD-cards at a comparable price, this can not be recommended.

Task 1: download a system image. To install an OS please download the system image provided in the data repository. If non of the software functions shown here are needed, one of the stock disk images (ether Ubuntu Linux or Android) from the ODroid wiki [13] can be used.[1] There is also a list of third party OS images not officially supported [22].

Task 2: copy the image to the eMMC. To copy the system image onto the eMMC an appropriate adapter for connecting it to a PC is needed. (We used the *Allnet debo eMMC 2 msd* purchased at *Reichelt GmbH & Co. KG* [23]).

[1] Refer to Section 6.4 for instructions on how to setup the gateway module's functionality without using the pre-preparred system image.

The open source tool "Etcher" [24] can be used to write the system image to the eMMC. It works on Windows, most Linux and Mac systems.

Caution: Make sure to select the correct destination (your eMMC), to prevent possible data loss!

Task 3: connect eMMC to the ODroid and power up the board. Now unmount the eMMC, unplug it from the adapter and connect it to the (not-powered!) ODroid.

Caution: If the eMMC has two connector strips make sure using the correct connector! Look for a little white dot next to the connector strip. For the right orientation put the dot on the eMMC and the dot on the ODroid on top of each other.

Power the board – it should boot up (red and blue LED on). After a while the blue LED will start flashing in an heartbeat pattern. This means the kernel was loaded successfully.

5.5. Network setup

If connecting to a DHCP-network just plug in an Ethernet cable and monitor your router for a device named "odroid" to obtain a DHCP lease.

Advice: To make administration easy it is recommended to add a static mapping for the ODroid's MAC address in your router. This makes sure the gateway module always gets the same IP address when connected to your network.

If you do not have a DHCP server in your network or do not have access to it to check the ODroids IP, connecting a screen and keyboard is necessary for a first setup of the networking. Log in with administrator login (user = *root*, password = *odroid*).

If you have a DHCP server and just want to check the IP address that was assigned to your ODroid type:

```
root@odroid:~# ip addr
```

5.5.1. Static IP setup

```
root@odroid:~# nmcli con add con-name "name" ifname eth0 type ethernet ip4 x.x.x.x/yy gw4 z.z.z.z
root@odroid:~# nmcli con mod "name" ipv4.dns "a.a.a.a,b.b.b.b"
root@odroid:~# nmcli con up "name"
```

If you want to configure a static IP address for the ODroid use the following commands:
Where:

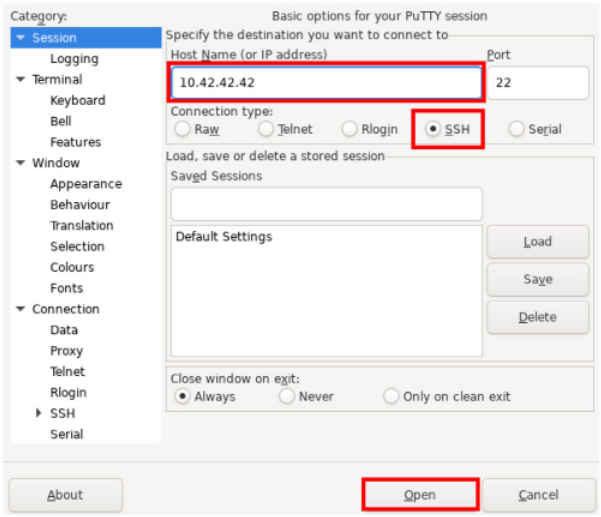
name would be the name of your new connection (just choose any you like)
x.x.x.x/yy would be the desired IP address and subnet mask in CIDR notation
z.z.z.z would be the IP address of your gateway (omit the gw4 parameter if you do not have one)
a.a.a.a,b.b.b.b would be the addresses of the DNS servers to use (use "8.8.8.8,8.8.4.4" for the Google DNS services)

5.6. Network connection to the board

Now that the networking is configured you can remote connect the board using the secure shell protocol (SSH). Most Linux systems have a SSH client pre-installed. If not, look for a package named something alike "ssh-client" in your distribution's package sources (i.e. "openssh-client" in Ubuntu/Debian). On Windows 10 you can use the build in SSH client from the shell (from Windows 10 version 1709 on) or the open source tool "Putty" [25].

Connect your client to the boards IP address with administrator login (user = *root*, password = *odroid*). When using the image provided by us a user named *user* is available (default password = *user*). In this example we assume the IP address to be *10.42.42.42*. On the first connection with a client, that was never logged in on the board before, you will have to confirm that you trust the connection.

In Putty select "SSH", type the IP in the "Host Name"-field and click "Open":



After that you will be prompted for user and password.
When using a a console-based ssh client type:

```
me@desktop:~$ ssh root@10.42.42.42
```

After that you will be prompted for the password.
When the board and network are working correctly you should be presented a bash prompt on the ODroid looking like this:

```
root@odroid:~#
```

5.7. Expanding the root-filesystem

If you did use the provided system image, first thing to do is expanding the root partition to the full size of your eMMC. For that, firstly delete your old root partition and create a new one with the same starting-block using the whole drive.

Task 1: run fdsisk. Use the *fdisk*-utility to do that.

```
root@odroid:~# fdisk /dev/mmcblk0
```

Replace "/dev/mmcblk0" with the block device that represents your drive. When using an ODroid C2 with an eMMC "/dev/mmcblk0" will most likely be correct.

Task 2: delete the old partition and create a new one. You will be presented with an interactive prompt. First press "p" to *print* the current configuration. You should see only one drive (your eMMC) with two partitions. The first one has an *fat* filesystem that holds */boot*. Do not touch this partition as the bootloader will otherwise stop working! The second partition is of a *linux* filesystem format and holds your root-filesystem. Press "d" to *delete* a partition and select the number of that partition (most likely "2"). Then press "n" for creating a *new* partition and select "p" to create a *primary* one. Choose the new partition to be at number "2" and use the defaults for first and last sector. The defaults will create a partition in the whole free space of your drive. You will be notified that the new partition contains an *ext4* filesystem signature and asked if you want to remove that. Do not remove the filesystem signature as you want to keep the data of the old (small) partition! You may use "p" again to check your new configuration (remember the name of the newly created partition, most likely "/dev/mmcblk0p2") and type "w" to write the changes to disk. *fdisk* will exit after successfully altering the partition table.

Task 3: reboot. Now reboot to load the new table:

```
root@odroid:~# reboot
```

Task 4: reconnect and expand the root-filesystem to the size of the new partition. After the ODroid is up again, establish a new SSH-connection and type:

```
root@odroid:~# resize2fs /dev/mmcblk0p2
```

Where "/dev/mmcblk0p2" is the device that represents your newly created partition.

Task 5: check for success. Your root filesystem is now expanded to use the entire drive. You can use the *df*-utility to confirm the result:

```
root@odroid:~# df -hT -xtmpfs -xdevtmpfs
```

5.8. Minimal version

What you have now is the Minimal Version of the gateway module. When you installed the system image provided by us you can go on using it as a SiLA2 server as described in Section 6.2.

This setup can for example be integrated into a small housing (i.e. the Bopla Gehäuse Systeme GmbH also provides different smaller cases) to serve different purposes in your lab network. Or when building a "house-made" device this setup can be integrated to serve as a small control unit and enable good connectivity for the lab device.

Common lab devices nowadays still offer a RS232 connector to be controlled. When you are planning to integrate such a device, a USB-to-serial converter can be used. However, these converters sometimes turned out to be unreliable. A better solution is to use the ODroid's on-board serial interface.

5.9. RS232 version

Unfortunately the on-board serial interface operates on a TTL-logic which raises the need for a converter board to comply with the RS232-standard normally used by labware manufacturers. A self-assembly kit is used as this makes exchanging components easy and thus allows for a flexible setup. Of course ready-build converter boards are available and can be used when this flexibility is not needed.

Task 1: assemble the RS232 converter board. The RS232 converter board is assembled according to its manual. The included IC (MAX232) must be changed to the MAX3232 as it will otherwise destroy the ODroids internal serial micro controller.

Caution: Even if the MAX3232 on the RS232 converter board can be supplied with 5 V the ODroid C2 is not 5 V tolerant. The supply voltage that is applied is given to the ODroid as ttl-voltage. To protect the ODroid, the RS232 converter board should be supplied with 3 V!

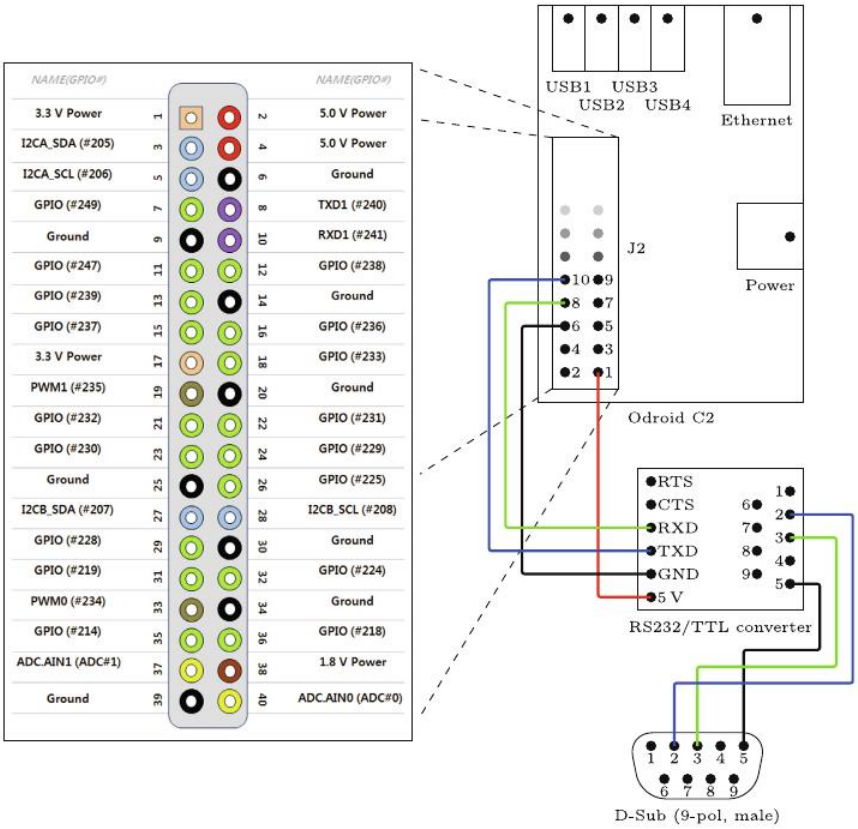
Caution: When soldering the converter chip please limit its heat uptake to a minimum to protect the circuits inside.

Task 2: solder the D-SUB connector. When a female D-SUB connector is needed it can be soldered directly to the board. For a male D-SUB connector (as the D-SUB 9-pole connector in our setup) the pinout of the connector will be mirrored. So the D-SUB pins and the converter pins with the same pin number must be connected by wire. For a D-SUB connector pin 2 is TXD, pin 3 is RXD and pin 5 is GND. In Fig. 5 a male D-SUB connector is shown.

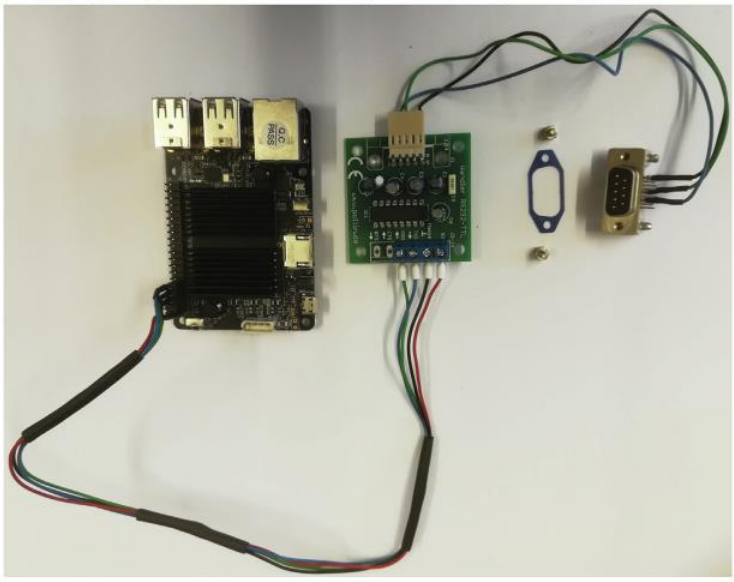
Caution: Do not solder a male D-SUB connector directly to the RS232 converter board!

Advice: There is no absolute convention on when to use a male or female D-SUB connector. However, male connectors are often used on the "computer-side" of a cable whereas female connectors are normally used on devices. Furthermore, when connecting devices often so called "null modem cables" with twisted wires are needed.

Task 3: connect the RS232 converter board to the ODroid. As can be seen in Fig. 5, the RS232 converter board is connected to the J2-expansion–header. Pin 1 of the J2-header (3,3 V) is connected to the 5 V pin of the RS232 converter board. The converter board will only need 3,3 V with the MAX3232. Pin 6 of the J2-header (GND) is connected to the GND pin of the converter board. Pin 8 of the J2-header (TXD1) is connected to the RXD pin of the converter board. Pin 10 of the J2-header (RXD1) is connected to the TXD pin of the converter board.

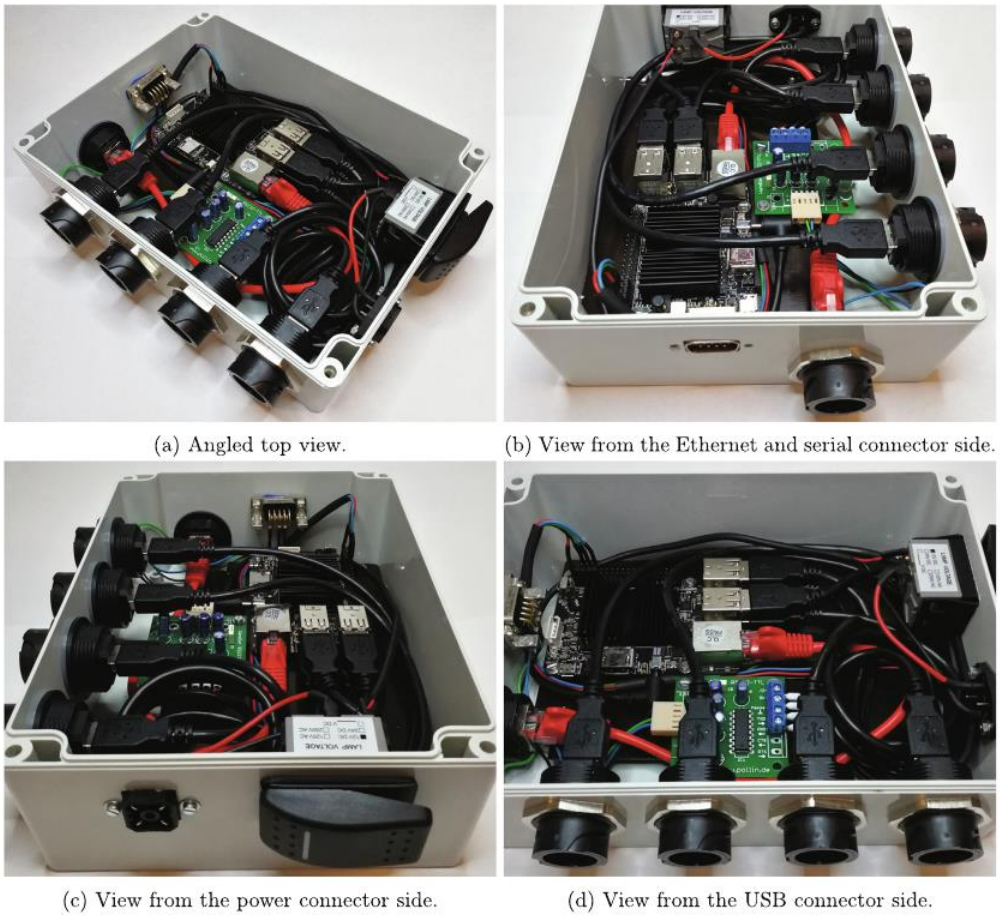


(a) Wiring diagram for connecting the RS232/TTL converter setup.



(b) Photo of complete serial connector assembly.

Fig. 5. Graphical instructions for assembling the RS232/TTL converter setup (Source of J2-header pinout drawing: [18]).



(a) Angled top view. (b) View from the Ethernet and serial connector side.

(c) View from the power connector side. (d) View from the USB connector side.

Fig. 6. Images of the assembled gateway module with the cover lid removed.

5.10. IP65 version

For applications that require spray water or dust protection the setup is housed into a Bopla M 223 G enclosure. Follow the Tasks stated below. For pictures of the completely assembled box see Fig. 6 and for a full wiring diagram refer to Fig. 7.[2]

Task 1: prepare the Bopla M 223 G enclosure. For external connections IP67 certified adapters are mounted into the Bopla M 223 G enclosure. For them drill or cut holes into the box as described in Fig. 8.

Caution: Please note that these connectors only offer IP67-certified shielding if the connected cables use the corresponding glands and if the unused connectors are protected by the corresponding blind caps!.

Task 2: prepare the base plate. Cut a piece of the Pertinax Hard Paper Plate to size and drill holes as shown in Fig. 9.

Task 3: mount the ODroid and RS232 converter board on the base plate. Assemble the baseplate as stated in Fig. 10 using the M2.5x12mm Screws, M2.5 Nuts and M2.5 Shims.

Advice: Make sure the eMMC is connected and the operating system is working as its connector will not be accessible when the ODroid is mounted onto the baseplate.

Task 4: mount the connectors into the Bopla M 223 G enclosure. Install the RJ45 Coupler, the four USB Couplers, the Belden Receptacle for the power connection, the Rocker Switch Assembly (see Section 6.3) and the D-SUB Connector in the Bopla M 223 G enclosure as shown in Fig. 11.

Advice: Make sure to use the Belden Seal when connecting the power connector to the receptacle and also the seal shipped with the D-SUB Connector.

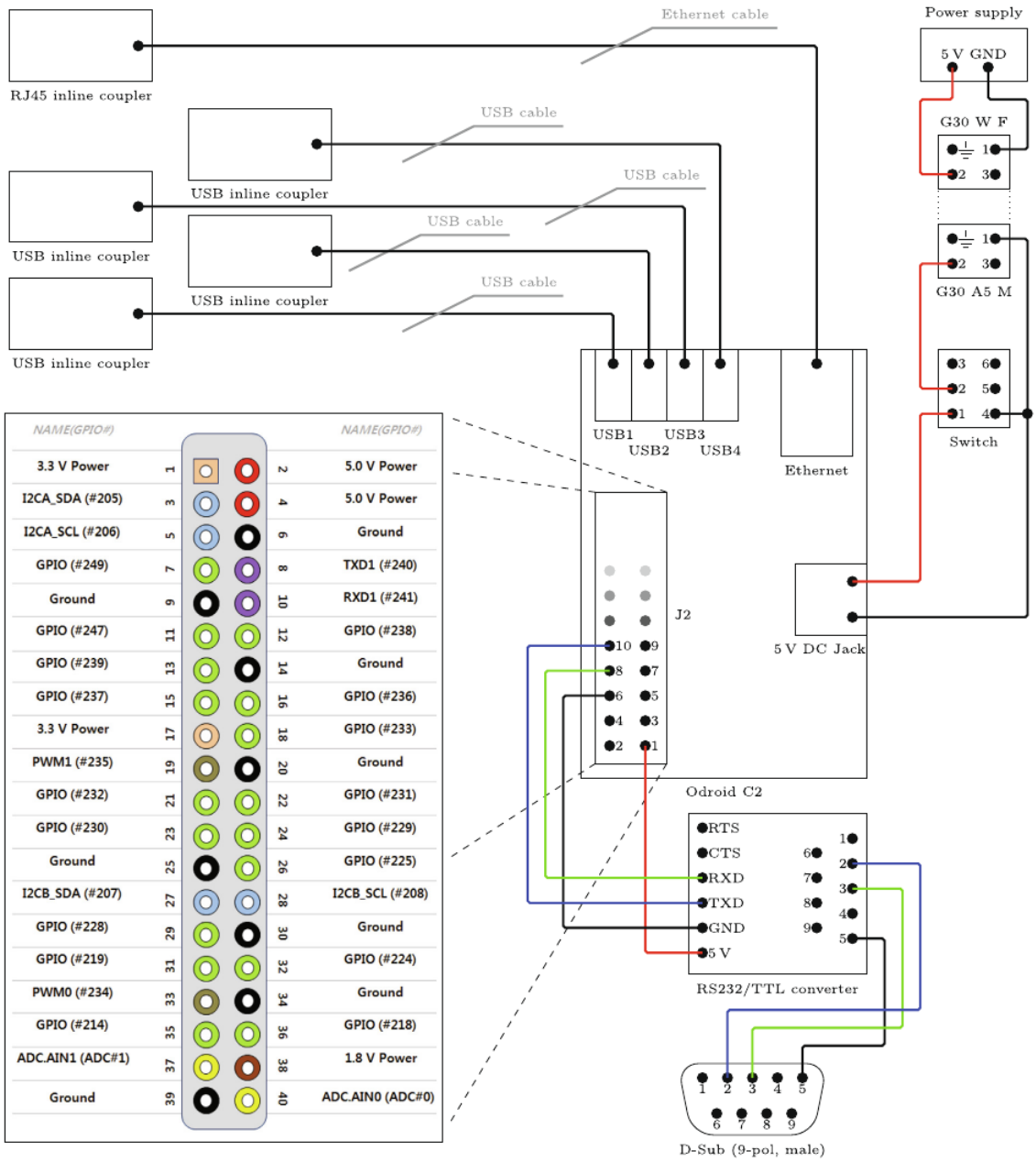


Fig. 7. Complete wiring diagram for the IP65 version (Source of J2-header pinout drawing: [18]).

Task 5: install the base plate, connect the connectors and mount the power assembly. Screw the assembled base plate to the bottom of the Bopla M 223 G enclosure using the three 3.9x9.5 Metal Screws and M4.3 Shims and connect the RS232 converter and the power assembly to the ODroid (see Fig. 12).

Connect the USB and RJ45 Couplers as can be seen in Fig. 13. Use the 0.5 m USB- or 0.25 m Ethernet-cables.

6. Operation instructions

In this section the setup and usage of the gateway module is described. If you want to start off with the pre-prepared system image to have a ready-to-operate gateway module with some example SiLA2 servers already installed, please follow the instructions in Section 6.2.

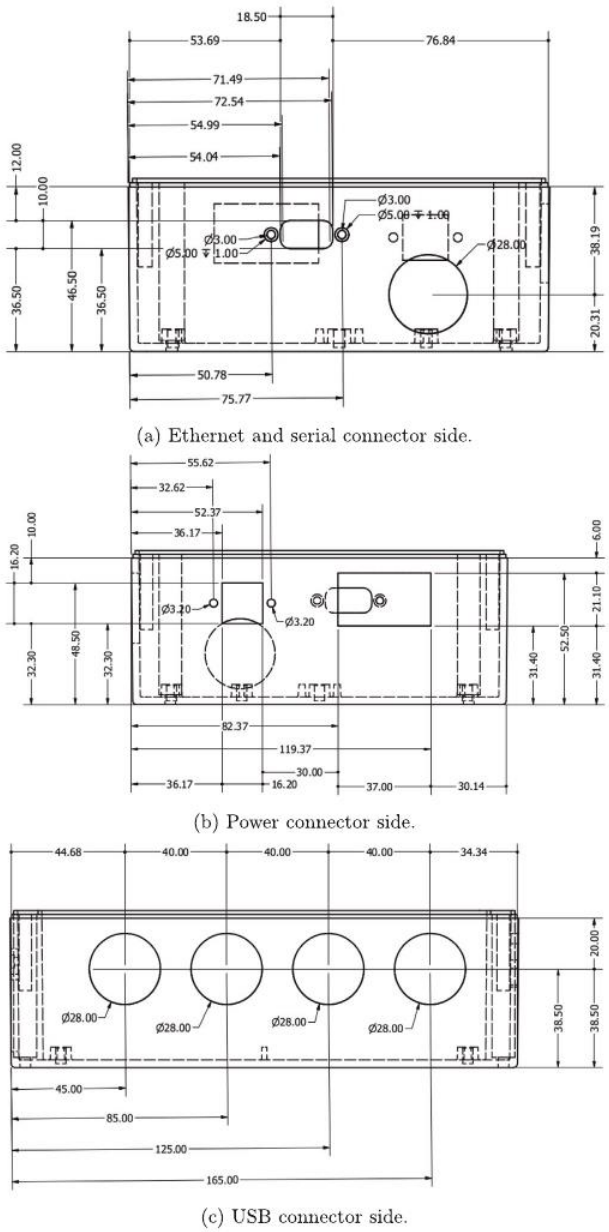


Fig. 8. Drawings of the holes that need do be machined into the Bopla M 223 G casing.

If you have the module running and want to add your own SiLA2 servers, follow the instructions in Section 6.3. Here the requirements needed on the desktop system used for development are also highlighted. In general *bash*-commands are used for easy description of procedures. They are either preceded by user@odroid to show that they are executed on the ODroid, or by me@desktop when they are to be ran on your desktop system.[3]

6.1. Potential safety hazards

Please use reasonable caution when operating electrical devices – especially in a potentially wet lab environment.

[3] This will work out of the box on Linux Systems, but also on Windows 10 with the "Windows Subsystem for Linux (WSL)" [26] installed.

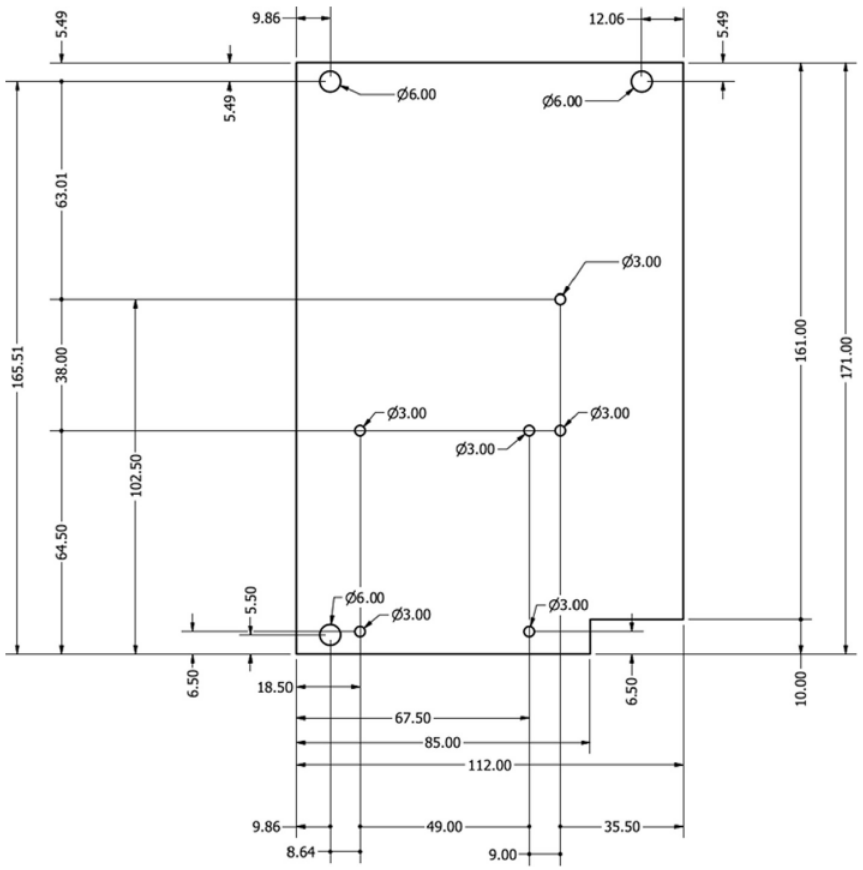


Fig. 9. Template for cutting and drilling the baseplate out of Pertinax Hard Paper Plate.

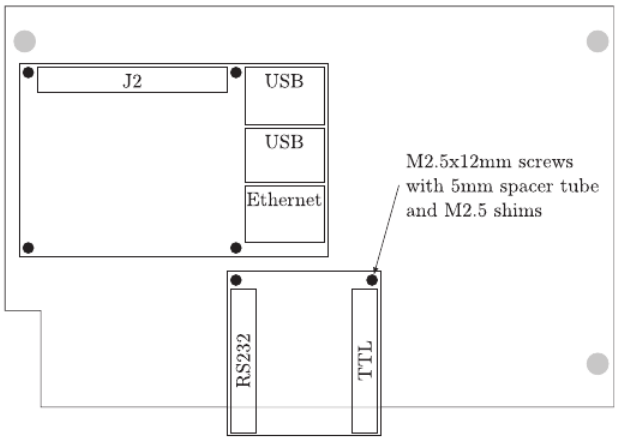


Fig. 10. Sketch of ODroid board and serial converter board positions on the basepl.

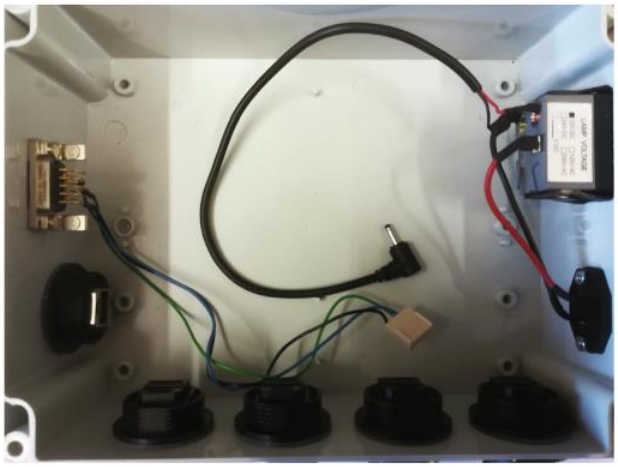


Fig. 11. RJ45 Coupler, four USB Couplers, Belden Receptacle for power connection, Rocker Switch Assembly and the D-SUB Connector mounted in the Bopla M 223 G enclosure.

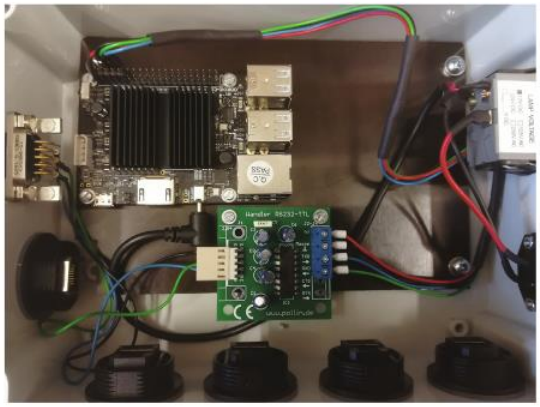


Fig. 12. The assembled baseplate screwed inside the box and connected to the serial converter and power supply.

6.2. Quick start

After setting up the gateway module with the prepared system-image it can be used as a SiLA2-server instantly. Please determine the module's IP-address as described in Section 5.5. Open a web-browser on any computer in the same network and surf to the module's IP (and default http-port:80). You are presented with the administration web frontend where you can control all SiLA2-servers currently installed on the gateway module.

To run the example server click the "start"-button next to it. The example server is a simple SiLA2-server presenting some commands and properties that showcase how SiLA2 works. It does not interact with any real lab device. You can use any SiLA2-client to connect to the server. For an easy start you can run the SiLA-browser. For instructions on how to get started with the SiLA-browser refer to [27]. Omit the part "Run a SiLA Server to Test the Browser with" as you already have your SiLA-server running on the gateway module. Please note that for automatic service discovery ("Scan Network") you network must be configured for mDNS. If service discovery is not working on your network you can always add the server by its IP-address and port. Use the administration web frontend to determine which server is running on which port. Now use the SiLA-browser to play around with your example server.[4]

[4] In the current version the SiLA browser is not displaying Intermediate Results from Observable Commands.

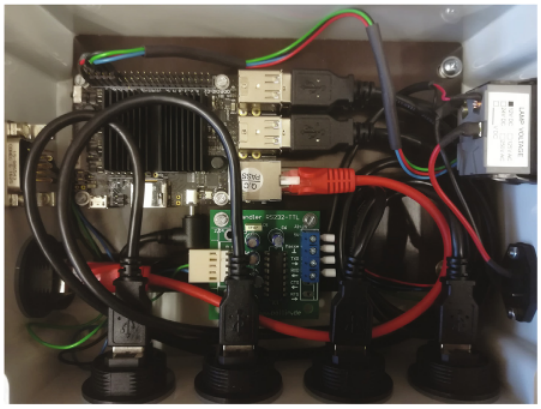


Fig. 13. The USB and Ethernet cables connected to the couplers.

For demonstration purposes a driver for a magnetic stirrer that only offers a very rudimentary serial interface and thus poses an ideal example of "making a dumb lab device smart" is also included. (This driver can be found in the git repository: [28]). If you happen to have a magnetic stirrer of the same type (*neoLab D-6010* or familiar) you can connect it to the RS232 serial port of the module using a nullmodem-cable (it will not work with a different cable!). After that, start and operate the stirrer's SiLA2-server the same way you did with the example.[5]

Advice: To change the default-password log in as *user* with password *user*. You can change this password with the command:

```
user@odroid:~$ passwd
```

6.3. Workflow for SiLA2-server development with the gateway module

The steps to develop a new SiLA2-Server and publish it to the gateway module are now briefly highlighted. This assumes that the module is set up ether by installing the provided system image or by following the steps in Section 6.4.

6.3.1. Software requirements (development system)

This section covers the requirements for your desktop development system. You will need:

- dotnet core SDK (v2.2 or higher) [29]
- recommended: C# IDE (i.e. Open Source Microsoft Visual Studio Code [30] with C#-plugin [31])
- GNU-Tools (like find, ssh, etc.): on any Linux machine you are fine. For Windows 10 machines preferably use the "Windows Subsystem for Linux (WSL)" [26].

6.3.2. Get the prerequisites on your development system

When developing a SiLA2 server on a desktop system, that is to run on the gateway module with ARM processor architecture, some extra prerequisites have to be met.

It is necessary for gRPC to be able to load ARM versions of the native library. To have these versions in place on the ODroid either use the prepared system-image or follow the steps in Section 6.4.2.

The gRPC implementation in the *arm-platform* branch of the gRPC fork is modified to dynamically load different native library versions based on the processor architecture on Linux systems. These precompiled library files can either be shipped with the application or can be installed in the system's library folders (as it is done here – see Section 6.4.2).

Task 1: clone the sila_tecan git repository. To use this mechanism when developing a SiLA2 server first clone the sila_tecan git repository [32] and switch to the branch *tci-gwm*:

[5] Please note that with the rudimentary API offered by neoLab no checking for correct command execution on the stirrer is possible. Even though the SiLA2-server tries to compensate by continuously monitoring the stirrer's parameters, it is possible that commands are sometimes not recognized correctly by the device.

```
me@desktop:~$ git clone https://gitlab.com/SiLA2/vendors/sila tecan
me@desktop:~$ cd sila tecan
me@desktop:~/sila tecan$ git checkout tci-gwm
```

Task 2: clone the forked gRPC git repository with modified library loading mechanism. The dependencies to gRPC in sila_tecan/Framework/Framework.csproj in the *tci-gwm* branch are configured in a way that it would normally install the official *Grpc.Core* NuGet package. But in case a clone of the gRPC repository is available next to the *sila_tecan* repo, it will instead use that version. This means you also need the *arm-platform* branch of the gRPC fork cloned next to *sila_tecan*:

```
me@desktop:~/sila tecan$ cd ..
me@desktop:~$ git clone --recurse-submodules https://gitlab.uni-hannover.de/tci-gateway-module/grpc
me@desktop:~$ cd grpc
me@desktop:~/grpc$ git checkout arm-platform
```

Task 3: build gRPC on the development system. As this alternative gRPC-implementation will now be used by the *sila_tecan* implementation, you should build the native library on the development system. Otherwise building and debugging your server on the desktop system will not be possible. For this task the gRPC documentation recommends to use the test python-script that will also build the native gRPC library [33]. You need to have cmake, Python and the Python-Library "six" installed for this to work:

```
user@odroid:~/grpc$ python tools/run tests/run tests.py -l csharp -c dbg --build only
```

Comment: On some systems some test may fail, which will lead to error messages after the *run_tests*-script finishes. However, as long as the native library (grpc/cmake/build/*grpc_csharp_ext.*) is build, everything will work.

6.3.3. Prerequisites summary

- use the *tci-gwm* branch in *sila_tecan* to be able to change the gRPC implementation used to the one modified to load ARM native libraries
- have a version of the C# native gRPC library build and installed on your ODroid (see Section 6.4.2)
- have the forked and modified gRPC git repo cloned next to *sila_tecan* (in a way that it is accessible from the *sila_tecan* root directory via *../grpc/*
- have the gRPC libraries build on your development system to be able to test and debug your server
- the server has to be implemented as a *netcoreapp{2;3}.** to run on the gateway module

6.3.4. Write your SiLA2 server application

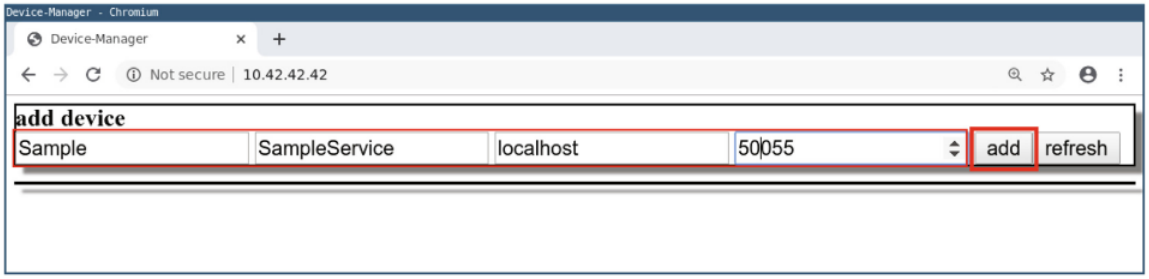
Now take a close look at sila_tecan/Examples/SampleServer and its readme [34] as it explains the steps to create a new server in detail. Follow these steps to create your new server. Make sure to create a dotnet core application as it will not run on the Linux system of the gateway module when you develop against the.NET Framework.

The SampleServer (Example from *sila_tecan*) will now be used to demonstrate the publish and remote control system. For productive purposes, exchange this to your newly written server.

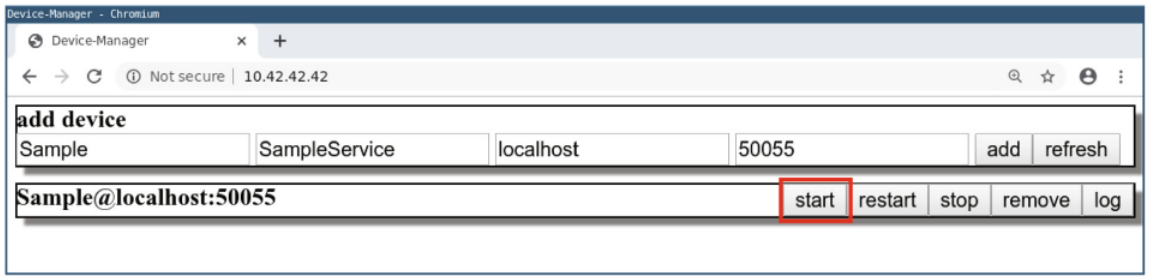
6.3.5. Publish the server to the gateway module

To use the automated publish system for the gateways first change to the cloned gateway-publish repo and make sure your gateway module is online and was prepared for remote publishing as described in Section 6.4.3 (if you use the prepared system-image this is all set up). It is assumed that the gateway has the IP address *10.42.42.42* and you want to publish the *SampleServer* project in /sila_tecan/Examples/SampleServer/SampleService/SampleService.csproj. When publishing, the *sila-servers* user that was created during setup is used. The publish system asks for the password of that user which is per default configured as *sila*.

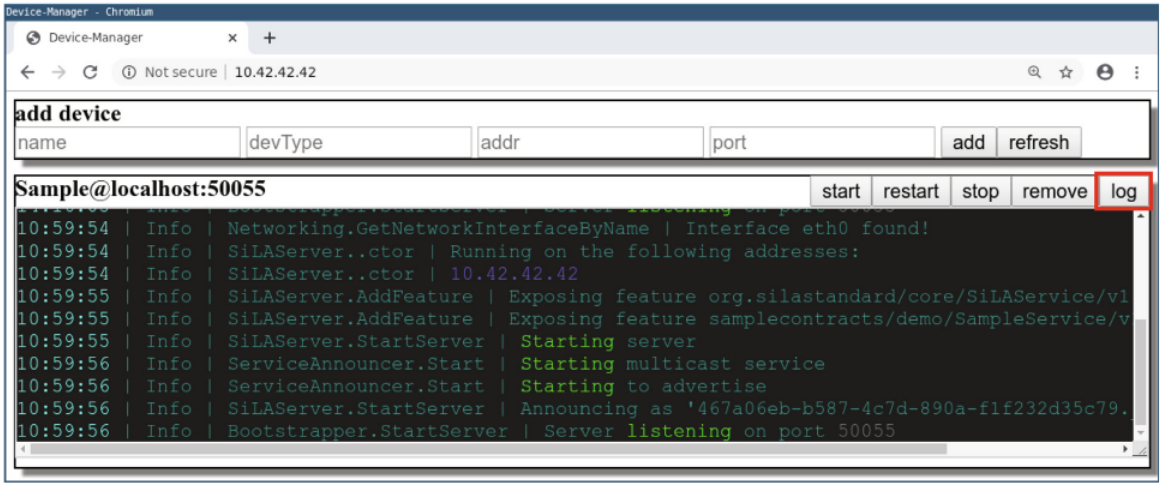
```
me@desktop:~$ git clone https://gitlab.uni-hannover.de/tci-gateway-module/gateway-publish
me@desktop:~$ cd gateway-publish
me@desktop:~/gateway-publish$ ./publish.sh 10.42.42.42 ../sila tecan/Examples/SampleServer/SampleService
        → /SampleService.csproj
```



(a) Use the "add device" pane of the Device-Manager web frontend to configure a new SiLA2 server.



(b) Use the "start"-button next to the newly added server to run it.



(c) The "log"-button expands a view of all logging by the underlying server application.

Fig. 14. Screenshots to demonstrate the usage of the Device-Manager web frontend.

When published this way, the SiLA2 server will start on all interfaces found. You can explicitly specify the network interface to run on by setting an environment variable *ip* while publishing. This variable can either contain the name of the interface or its IP address in CIDR notation.

For example using *ip="eth0"* will configure the service to run the server on the wired network interface of the ODroid explicitly:

```
me@desktop:~/gateway-publish$ ip="eth0" ./publish.sh 10.42.42.42 <path to csproj>
```

Alternatively using *ip="x.x.x.x/yy"* will setup the server to run on the interface with the specified IP address:

```
me@desktop:~/gateway-publish$ ip="10.42.42.42/24" ./publish.sh 10.42.42.42 <path to csproj>
```

6.3.6. Remote control the SiLA2 server with the Device-Manager web frontend

The *Device-Manager* web frontend can be used to remote control and debug the server. Setup and run the *Device-Manager* on the gateway module (or your dedicated server) as stated in Section 6.4.4. If you use the prepared system-image it is already running on port:80 of the gateway module.

Open a web browser and surf to the address the manager is running on. Now you may add your new SiLA2 server. Refer to Fig. 14a and use the "add device" pane (see Fig. 14a). In the first field enter a descriptive name, in the second use the name of the *.csproj* file of your server. In the third and fourth specify the gateway module's IP and the port the server should be started on. If you use the manager frontend on the gateway module itself, you can simply use "localhost" insted of the IP. By clicking "add" a new entry in the list below will be added for your SiLA2 server.

When adding a server on a new device, login credentials must be specified once. This is possible with the password field. Enter the password of the "–silaUser" (default: "sila-servers") on the target device and press the login button.

Now run the newly configured server by clicking the "start" button next to it (see Fig. 14b). This will run the systemd service. For debugging purposes you can use the "log" button for expanding a logging view of the dotnet process (see Fig. 14c).

For running commands on your newly configured SiLA2 server, any SiLA2 compatible client can be used. You could ether write one yourself[6] or use an existing one. For testing purposes the *SiLA-Browser* is a handy tool. Use it as described in Section 6.2.

For a detailed description on how to operate the *Device-Manager* (including steps to install and run it on a dedicated server and using configuration files for easy setup of SiLA2 server lists) refer to its manual [35].

6.3.7. Remote control the SiLA2 server "by hand"

During publishing a systemd-service for the new SiLA2 server was installed on the gateway module. This can be used to remote control the server "by hand" (without using the web frontend) when necessary. The publish system specifies the port on which the server should run with its parameter *-p*. The template service-file (*server@.service.in*) is configured in a way that the parameter of the service (given in the command after the @-character) is forwarded to the SiLA2 server as the *-p*-Parameter. To run the server on port 50 055 use:

```
me@desktop:~$ ssh sila-servers@10.42.42.42 systemctl start SampleService@50055.service
```

6.4. Detailed setup instructions

This section will clarify how to setup the module without using the pre-prepared system-image. Please follow the Tasks stated below and modify/omit/extend the steps to your needs.

6.4.1. Operating system installation

Task 1: choose a system image. Download an ODroid C2 system image from either the official source [13] or use a third party image [22]. For any further steps we assume the *ubuntu-18.04.3–3.16-minimal-odroid-c2-20190814.img.xz* image is used. If you opt for another one, please adapt the procedures accordingly.

Task 2: install the base system to the eMMC. Ether follow the steps from Section 5.4 (using *Ether*) or use any other tool for device dumping (i.e. *dd* on Unix systems) to copy the system image to your eMMC or SSD-card:

```
me@desktop:~$ xzcat ubuntu-18.04.3-3.16-minimal-odroid-c2-20190814.img.xz | sudo dd of=/dev/sdX status
        → =progress
```

Caution: Replace */dev/sdX* with your eMMC path. Make sure to select the correct destination, to prevent possible data loss!.

Task 3: boot up the ODroid and configure the network connection. Connect the eMMC to the ODroid, boot it up, check and configure its networking (see Section 5.5) and establish a SSH-connection as described in Section 5.6.

Task 4: add a user. For all following instructions we assume that you have a user named "user" on your ODroid. You can add this user with the command:

```
root@odroid:~# adduser user
```

[6] There are several SiLA2 implementation available that all offer methods of writing clients (see [12]). I.e. the sila_tecan implementation can be used for developing SiLA2 clients as well.

You also want to add this user to the sudoers group to give it permissions to perform administrative tasks:

```
root@odroid:~# usermod -a -G sudo user
```

If you are planning to use the gateway module with serial- or USB-devices, the newly created user should be added to the dialout group to gain rights to communicate with such devices:

```
root@odroid:~# usermod -a -G dialout user
```

Task 4: expand the root filesystem if necessary. Depending on the system image you used, the *root*-partition may not be using the whole size of your drive. Apply the steps described in Section 5.7 to expand the partition.

Task 5: Login with the new user. Now log off the "root" user (*exit*) and back on using the newly created user "user".

Task 6: update the system. Before going on it is recommended to update your system. As of January 2020 you will have to update a PGP-Key first, as the one in the minimal Ubuntu image is expired.

```
user@odroid:~$ sudo apt-key adv --keyserver keyserver.ubuntu.com --recv-keys AB19BAC9
user@odroid:~$ sudo apt update
user@odroid:~$ sudo apt dist-upgrade
```

Advice: It is likely that the *dist-upgrade* installed a new Linux-kernel version. Reboot the box to run this new version:

```
user@odroid:~$ sudo reboot
```

Task 7: install software. The minimal Ubuntu image comes only with a very lightweight set of software.
Suggestion: In case you can spare some more megabytes, installing a few extra tools will make you life a lot easier:

```
user@odroid:~$ sudo apt install man bash-completion usbutils screen git psmisc htop vim pydf
```

Advice: Reload the *.bashrc* to enable the bash-completion scripts that make using the shell a lot easier (providing tab-auto-completion for a great set of commands):

```
user@odroid:~$ source .bashrc
```

Task 8: setup automatic filesystem checks. When in day to day use clean shutdowns of embedded hardware can not always be guaranteed.
Advice: To setup automatic file system checks and repairs on every boot use the *tune2fs*-utility to set the maximum mount count of the root partition to 1:

```
user@odroid:~$ sudo tune2fs -c 1 /dev/mmcblk0p2
```

6.4.2. gRPC for C# Setup (on aarch64 Systems)
Unfortunately Google is not offering compiled gRPC executables for ARM-processor-architectures (used in most single-board-computers). This means that some additional steps have to be taken to run SiLA2. First compiling the gRPC native library for aarch64 (ARM 64Bit) architecture is necessary. Second (in case of dotnet C#) also modifying the C#-bindings to load this new library needs to be done.
These changes are available at the moment only in a repository that was forked by the authors from the official gRPC GitHub-repo [36]. When using the *sila_tecan* SiLA2 implementation this repo needs to be available during build-time for SiLA2 to work on ARM systems. Refer to Section 6.3 for detailed instructions. Also before starting to write a SiLA2 server,

a compiled gRPC library for ARM should be available on the target system. You can either use the one provided with this article's supplementary data or compile it on your own ODroid.

Task 9: clone the forked gRPC git repository. For compiling it yourself please also use the forked gRPC repository (as some changes were needed for gRPC to successfully build on ARM) and switch to branch *arm-platform*:

```
user@odroid:~$ git clone --recurse-submodules https://gitlab.uni-hannover.de/tci-gateway-module/grpc
user@odroid:~$ cd grpc
user@odroid:~/grpc$ git checkout arm-platform
```

Task 10: install gRPC dependencies and build the gRPC native bindings library. First install all dependencies and use cmake to build the native C# extension library:

```
user@odroid:~/grpc$ sudo apt install build-essential autoconf libtool pkg-config libunwind-dev golang-go
        → cmake
user@odroid:~/grpc$ mkdir cmake/build
user@odroid:~/grpc$ cd cmake/build
user@odroid:~/grpc/cmake/build$ cmake ../..
user@odroid:~/grpc/cmake/build$ make -j5 grpc csharp ext
```

Task 11: copy the newly build library to the ODroid system's library sources. Now copy the library to the system's library sources and rename the shared library file in a way that the modified loading system of the gRPC–C# binding library can automatically detect it:

```
user@odroid:~/grpc/cmake/build$ sudo cp libgrpc csharp ext.so /usr/lib/libgrpc csharp ext.aarch64.so
user@odroid:~/grpc/cmake/build$ sudo ldconfig
```

6.4.3. Prepare the publish-system with dotnet core C# environment setup

To host a SiLA2 server on the gateway module the open source sila_tecan implementation was chosen [32]. This C# library offers a convenient method of writing SiLA2 servers and was ported to dotnet core by the authors to be usable on Linux machines. These modifications are to be found in the official sila_tecan repository [32]. Using dotnet for SiLA2 server development offers the possibility of pre-compiling code on desktop-systems with architectures differing from the one of the ODroid. In combination with the presented publish- and control-system this ensures flawless development and debugging.

Alternatively any other SiLA2-implementation [12] can be used as well. Refer to Section 6.4.2 and adapt the steps there according to our language of choice.

To install SiLA2 servers on the gateway-module use the provided script that can be found in a git-repository [37]. It compiles a CIL-version of the dotnet core application and packs it for publishing. This runtime-executable version is then transferred to the module via SSH-Filesystem and a system-service is registered with the module's systemd-system. A user *sila-servers* is created during setup to control all SiLA2 servers with appropriate rights. The systemd-service can be used for remote-controlling the SiLA2 server, either directly (via SSH) or – more convenient – with the *Devie-Manager* web-frontend.

For using C# a dotnet runtime for aarch64 is necessary on the ODroid. The publish system comes with an installation script that also sets up a dotnet core runtime.[7]

Task 12: clone the publish system git repository on the development machine. To set up the gateway module for remote publishing, clone the git repository:

```
me@desktop:~$ git clone https://gitlab.uni-hannover.de/tci-gateway-module/gateway-publish
me@desktop:~$ cd gateway-publish
```

[7] If you do not want to install the publish system but need dotnet core, download and install the tarball provided by Microsft and following the instructions [38] (This link refers to.NET Core 3.1 for aarch64. Also different versions and other architectures are available).

Task 13: install the publish system on the ODroid. Then execute the setup-script on the ODroid (replace the IP-address with the one of the ODroid in your network):

```
me@desktop:~/gateway-publish$ ssh root@10.42.42.42 "bash -s" <./setup.sh
```

For a detailed description and instructions refer to the readme in the gateway-publish git repository [39].

6.4.4. Install the device-manager web frontend

A web frontend (called *Device-Manager*) for remote administration and debugging of all SiLA2 servers running on one or more gateway-modules was developed. The systemd-services created by the publish-script are remote controlled via SSH. Also all logs from the SiLA2-servers are collected and presented in the web frontend. (Re-) starting or stopping of all SiLA2-servers in the network can be performed as well. This software is available in a git-repository [40].

The *Device-Manager* can run on the module itself (as it does when using the provided system-image) or can be hosted by a dedicated server. The latter is useful when several gateways are in use in a network and a system of convenient central administration of all of these is necessary. If you want to install it on your own server please follow the instructions in the readme of the repository [35].

Task 14: install the device-manager web frontend's dependencies on the ODroid. To install the *Device-Manager* on the gateway module, a D building environment is needed. On ARM architectures the ldc can be used. Please install all its dependencies and download the tarball:

```
user@odroid:~$ sudo apt install sshpass ccze expect zlib1g-dev libssl-dev
user@odroid:~$ wget https://github.com/ldc-developers/ldc/releases/download/v1.20.1/ldc2-1.20.1-linux-
        → aarch64.tar.xz
user@odroid:~$ tar xfv ldc2-1.20.1-linux-aarch64.tar.xz
```

Task 15: configure a swap-file. The build process will need more RAM than the 2 GB the ODroid C2 has to offer. So please configure a swap file:

```
user@odroid:~$ sudo fallocate -l 2G /swap
user@odroid:~$ sudo chmod 600 /swap
user@odroid:~$ sudo mkswap /swap
user@odroid:~$ sudo swapon /swap
```

Task 16: Clone the device-manager git repository and use the ldc to build it. Now clone the *Device-Manager* repository and use the ldc to build it:

```
user@odroid:~$ git clone https://gitlab.uni-hannover.de/tci-gateway-module/device-manager
user@odroid:~$ cd device-manager
user@odroid:~/device-manager$ ../ldc2-1.20.1-linux-aarch64/bin/dub build
```

Task 17: remove the swap-file. After building you may remove the swap file:

```
user@odroid:~device-manager$ sudo swapoff /swap
user@odroid:~device-manager$ sudo rm /swap
```

Task 18: Install the device-manager. To install a systemd service, copy the compiled executable and the prepared service-files to an appropriate location:

```
user@odroid:~device-manager$ sudo mkdir -p /opt/device-manager/bin
user@odroid:~device-manager$ sudo cp -r device-manager public /opt/device-manager/bin/
user@odroid:~device-manager$ sudo cp *.service /etc/systemd/system/
user@odroid:~device-manager$ sudo systemctl daemon-reload
```

Task 19: setup a configuration for the device-manager and use systemd to run (enable) it. To run the *Device-Manager* with a pre-configured set of SiLA2 servers in the list you may add a configuration file. As an example the *exampleConfig.json* from the repository is used here:

```
user@odroid:~device-manager$ sudo cp exampleConfig.json /opt/device-manager/bin/
```

To control a systemd service created by the publish system the *Device-Manager* needs the following information:

name identifier that uniquely refers to this service
devType name of the service file (same as the.csproj file)
addr network address of system hosting the service. This can either be "localhost" (in case the SiLA2 server is installed on the same system as the *Device-Manager*) or any IP address on the network
port port the SiLA2 server should be running on (forwarded from the systemd service file to the *-p* parameter of the dot-net program)

A configuration file needs to hold these information as a json-array. For example the *exampleConfig.json* looks like this:

```
[
        ["Sample", "SampleService", "localhost", 50055],
        ["Magneto", "MagnetoService", "localhost", 50056]
]
```

This config sets up two SiLA2 services (that have to be installed with the publish system beforehand – see Section 6.3) to run on the gateway module on ports 50 055 and 50 056.

To enable the *Device-Manager* (which will result in it automatically starting on system boot) with this configuration run:

```
user@odroid:~device-manager$ sudo systemctl enable "device-manager@exampleConfig.json"
```

To run the *Device-Manager* right away, start the systemd service:

```
user@odroid:~device-manager$ sudo systemctl start "device-manager@exampleConfig.json"
```

6.4.5. Final remarks

After following the above guidelines the setup of the module should be exactly the same as in the prepared system image. To use the module for hosting your own SiLA2 server follow the guide in Section 6.3.

Please be advised that the procedure outlined here is valid for the described HEADs of git branches at the time of writing. "Snapshots" of these repositories are also available in the supplementary data repository associated with this article. For updated instructions on the installation procedures, that also take into account future changes of the described software, please refer to the documentation in the repositories themselves.

7. Validation and characterization

In this sections the results of some testing procedures that were carried out to proof the reliability and durability of the setup are shown. For all tests the fully assembled IP65 version of the gateway module was used. Also a "real life" application is shown, in which the gateway module is being used as a SiLA2 server for a magnetic stirrer.

7.1. Heavy load test

To show that operating the ODroid in an completely enclosed housing without an active cooling system is not critical, a heavy load test was carried out. The fully assembled gateway module was placed in a room where the temperature was continuously between 18°C and 21°C, which should reassemble the conditions in most laboratories. Using the *stress*-utility a constant load of 100% was put on all four processor cores of the ODroid inside the module. During a timespan of 130m the reading of the internal temperature sensor of the ODroid was monitored. The results are shown in Fig. 15.

The CPU temperature rises from an idle value of approximately 47°C and reaches a maximum of 64°C after 30m which is never exceeded throughout the whole duration of the experiment. The ODroid user manual states 85°C as a maximum operation temperature of the CPU (p.13 [41]), which leads to the conclusion that even in the completely enclosed housing of the IP65 version high load operations are safe to perform.

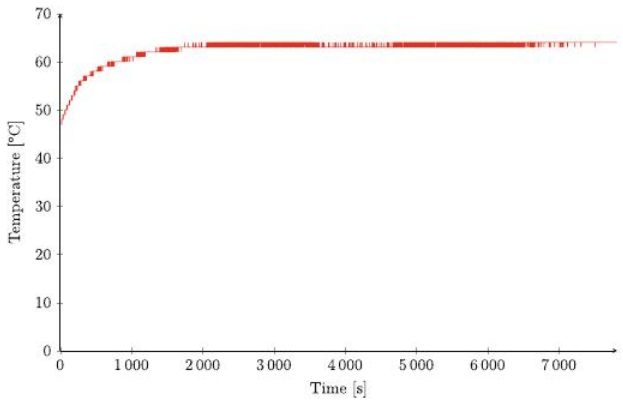
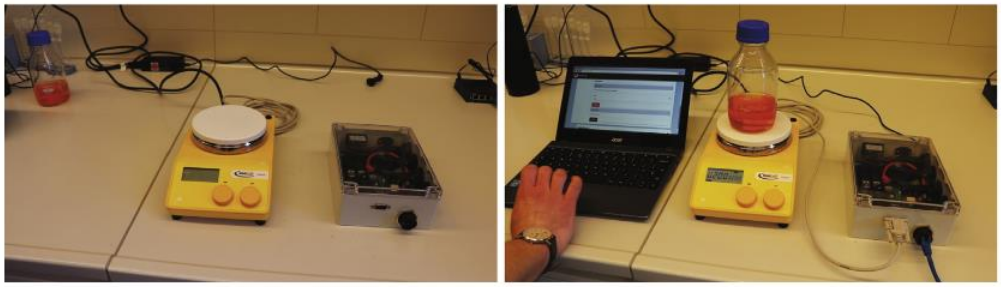


Fig. 15. Temperature readings from the ODroids internal thermal sensor during stress test.



(a) Preparation of the module. (b) Presentation of the module in use.

Fig. 16. Usecase examples of the gateway module used to act as a SiLA2 server for a *neoLab D-6010* magnetic stirrer. A: Preparation of the module (Link to video: https://data.mendeley.com/datasets/fvccdd6r4f/1/files/6eb9763f-b6e0-4a8d-a01b-7c916958a7f4/magnetosetup.mp4?dl=1), B: Presentation of the module in use (Link to video: https://data.mendeley.com/datasets/fvccdd6r4f/1/files/f125e65c-a726-4690-b7bb-e72aeec1b633/magnetocontrol.mp4?dl=1).

7.2. 24/7 Service test

To test the gateway module in a simulated long-term service situation and to ensure all components are working reliably when in 24/7 use, the SiLA2 server for the magnetic stirrer was ran continuously for one week (168 h) while a SiLA2 client was calling commands. The stirring was alternately switched on with 1000 rpm and switched off with a 2 s delay between the finishing of the commands.

The test showed that none of the components used for the gateway module was affected in any way by this continuous usage. Thus it is possible to conclude that the hardware and software presented is capable of running in 24/7 service for longer periods of time.

7.3. Use case presentation

The gateway module was used as described in Section 7 with a *neoLab D-6010* magnetic stirrer to showcase the practical functionality as a SiLA2 gateway. The SiLA-browser [27] was used to access and remote control the device via the network. Use the links in Fig. 16a and Fig. 16b to access the videos in the supplementary data repository. The first video shows and describes the preparations of the gateway module to act as a SiLA2 server for the *neoLab D-6010* magnetic stirrer whilst the second video shows the steps of operation.

7.4. Capabilities and limitations

Capabilities

- Universal device integration for the lab (for example with SiLA2)
- Network controllable

- Easy development and debug pipeline when used with SiLA2 and the tools provided
- Centralized device management/monitoring with the *Device-Manager* web frontend
- Open and free software running on a cost-efficient hardware design
- Reliable hardware and bullet proof Linux system allows 24/7 service
- Enables automated in-time data-acquisition (for example for FAIR-Data [5] compliant use-cases).

Limitations

- Only one RS232 serial port in the presented design (more are possible with the ODroid C2)
- Processing power of embedded computer solutions is limited (even though the powerful ODroid C2 is used)
- Usage of gRPC (needed by SiLA2) on ARM-architectures requires manual compilation of the library at the moment (no precompiled dotnet NuGet packages for ARM-architectures) and some changes in the dotnet bindings (see Section 6.4.2) are necessary

8. Conclusions

Many laboratories are now facing the challenge of digitizing workflows and data acquisition mechanisms. In this process devices that are working perfectly fine often need to be exchanged for newer ones with better connectivity capabilities. In this article a hardware module was presented, discussed and validated that can be used to integrate legacy devices into a digital lab infrastructure.

It could be shown that it is possible to achieve cost efficient digital integration with standard conform device communication and data structures. The presented gateway module – in contrast to commercially available solutions – is dust and spray-water proof, facilitates reliable hardware components and thus is ideal for the day to day laboratory use.

The software presented enables the researcher to easily develop a device communication infrastructure that is compliant to the open source SiLA2 standard. A complete toolchain for creating, debugging and publishing SiLA2 servers for the gateway module is made available as open source software. Common and widespread open source tools are utilized – such as a Linux operating system, the dotnet core C# programming language and gRPC for network communication.

The solution presented is flexible and scalable. On the one hand, one gateway module can be used to integrate several lab devices. On the other hand, multiple gateway modules can easily integrate into a common infrastructure and the presented *Device-Manager* centralizes administering and controlling SiLA2 servers running on all modules. The *Publish-System* ensures a straightforward integration workflow from software development on a desktop system to testing and running on the gateway modules.

Declaration of Competing Interest

The authors declare that they have no known competing financial interests or personal relationships that could have appeared to influence the work reported in this paper.

Human and animal rights

No humans subjects or animals were involved in the studies that lead to the publication of this article.

Acknowledgements

The authors thank the German Federal Ministry of Education and Research for funding our research in context of the project "Digitalisierung in der Industriellen Biotechnologie (DigInBio)" (BMBF FKZ 031B0463C) and the Projektträger Jülich.

The authors also thank their partners in that project: the Institut für Bio- und Geowissenschaften, Department for Bioprocesses and Bioanalytics at the Forschungszentrum Jülich and the Faculty of Mechanical Engineering, Institute of Biochemical Engineering at the Technical University of Munich for testing prototypes of the hardware presented here.

They also thank the Tecan Software Competence Center for help and support with their open source SiLA2 implementation and the Bopla Gehäuse Systeme GmbH for permission to publish CAD-Files and drawings based on their technical documentation under an open hardware license.

The publication of this article was funded by the Open Access fund of Leibniz Universität Hannover.

References

[1] T. Chapman, Lab automation and robotics: automation on the move, Nature 421 (6923) (2003) 661–666, https://doi.org/10.1038/421661a.
[2] G. Gauglitz, Lab 4.0: SiLA or OPC UA, in: Analytical and Bioanalytical Chemistry 410.21 (2018), pp. 5093–5094. doi: 10.1007/s00216-018-1192-6..
[3] I. Schmid, J. Aschoff, A scalable software framework for data integration in bioprocess development, Eng. Life Sci. 17 (11) (2017) 1159–1165, https://doi.org/10.1002/elsc.201600008.
[4] D.S. Lütjohann, N. Jung, S. Bräse, Open source life science automation: design of experiments and data acquisition via dial-a-device, Chemometr. Intell. Lab. Syst. 144 (2015) 100–107, https://doi.org/10.1016/j.chemolab.2015.04.002.

[5] M.D. Wilkinson et al, The FAIR Guiding Principles for scientific data management and stewardship, Sci. Data 3 (1) (Dec. 2016), https://doi.org/10.1038/sdata.2016.18 160018.
[6] S. Karnouskos, T. Bangemann, C. Diedrich, Integration of legacy devices in the future SOA-based factory, vol. 13. PART 1. IFAC, 2009, pp. 2113–2118. doi: 10.3182/20090603-3-RU-2001.0487.
[7] J.G. Frey, Dark lab or smart lab: the challenges for 21st century laboratory software, Org. Process Res. Dev. 8 (6) (2004) 1024–1035, https://doi.org/10.1021/op049895g.
[8] C. Stephan et al, Using laboratory information management systems as central part of a proteomics data workflow, Proteomics 10 (6) (2010) 1230–1249, https://doi.org/10.1002/pmic.200900420.
[9] D.O. Skobelev et al, Laboratory information management systems in the work of the analytic laboratory, Measure. Tech. 53 (10) (2011) 1182–1189, https://doi.org/10.1007/s11018-011-9638-7.
[10] E. Dolgin, Labs on the cheap, Nature 559 (7713) (2018) 291–293, https://doi.org/10.1038/d41586-018-05655-3.
[11] Association Consortium Standardization in Lab Automation (SiLA). SiLA2 wiki with links to all SiLA2 specification docments. url: https://gitlab.com/SiLA2/sila base/-/wikis/home/ (visited on 01/21/2020)..
[12] Association Consortium Standardization in Lab Automation (SiLA). SiLA2 Repository. url: https://gitlab.com/SiLA2/ (visited on 01/31/2020)..
[13] Hardkernel co., Ltd., ODroid wiki – getting started with ODroid C2. url: https://wiki.odroid. com/odroid-c2/getting started/os installation guide?redirect=2 #tab odroid-c2 (visited on 01/21/2020)..
[14] Labforward GmbH. LabOperator home page. url: https://www.laboperator.com/ (visited on 01/21/2020)..
[15] UniteLabs AG. UniteLabs AG home page. url: https://unitelabs.ch/ (visited on 01/21/2020)..
[16] UniteLabs AG. SiLA 2 Converter product page. url: https://sila-standard.com/dipitems/sila-2-converter-by-unitelabs/ (visited on 01/21/2020)..
[17] Bopla Gehäuse Systeme GmbH. Bopla M233 G technical data. url: https://www.bopla.de/gehaeusetechnik/product/euromas-abs-pc-f05/euromas-gehaeuse-pc/m-223-g.html (visited on 01/23/2020)..
[18] Hardkernel co., Ltd. url: https://wiki.odroid.com/odroid-c2/hardware/hardware (visited on 01/23/2020)..
[19] P.J. Basford et al, Performance analysis of single board computer clusters, Fut. Gen. Comput. Syst. 102 (2020) 278–291, https://doi.org/10.1016/j.future.2019.07.040.
[20] Hardkernel co., Ltd. ODroid C2 description. url: https://www.hardkernel.com/shop/odroidc2/(visited on 01/21/2020)..
[21] Jeff Geerling. Review: ODROID-C2, compared to Raspberry Pi 3 and Orange Pi Plus. url: https://www.jeffgeerling.com/blog/2016/review-odroid-c2-compared-raspberry-pi-3-andorange-pi-plus (visited on 01/24/2020)..
[22] Hardkernel co., Ltd. ODroid wiki - third party images. url: https://wiki.odroid.com/odroidc2/os images/third party (visited on 01/23/2020)..
[23] Reichelt GmbH & Co. KG. ALLNET DEBO EMMC 2 MSD. url: https://www.reichelt.de/entwicklerboards-emmc-zu-microsd-debo-emmc-2-msd-p248580.html?r=1 (visited on 01/23/2020)..
[24] Balena. Etcher download page. url: https://www.balena.io/etcher/ (visited on 01/23/2020)..
[25] Putty. Putty download page. url: https://www.putty.org/ (visited on 01/23/2020)..
[26] Microsoft. Windows Subsystem for Linux Installation Guide for Windows 10. url: https://docs.microsoft.com/en-us/windows/wsl/install-win10 (visited on 03/10/2020)..
[27] Timothy Diguiet. SiLA Browser Quickstart. url: https://gitlab.com/SiLA2/sila base/-/wikis/SiLA%20Browser%20Quickstart (visited on 01/27/2020)..
[28] TCI. SiLA2 Server for neoLab D-6010 magnetic stirrers. url: https://gitlab.uni-hannover. de/tci-gateway-module/magnetosiladriver (visited on 02/24/2020)..
[29] Microsoft. NET Core. url: https://dotnet.microsoft.com/download (visited on 02/04/2020)..
[30] Microsoft. Visual Studio Code. url: https://code.visualstudio.com/download (visited on 02/04/2020)..
[31] Microsoft. Visual Studio Code C Sharp Extention. url: https://marketplace.visualstudio.com/items?itemName=ms-vscode.csharp (visited on 02/04/2020)..
[32] Tecan Group. Tecan SiLA2 SDK repository. url: https://gitlab.com/SiLA2/vendors/sila tecan (visited on 01/21/2020)..
[33] Google. gRPC Contributing and Building Guide. url: https://github.com/grpc/grpc/blob/master/CONTRIBUTING.md (visited on 02/28/2020)..
[34] Tecan Group. Tecan SiLA2 SDK SampleServer Readme. url: https://gitlab.com/SiLA2/vendors/sila tecan/-/blob/master/Examples/SampleServer/Readme.md (visited on 02/04/2020)..
[35] TCI. Installation and Operation Instructions: Device-Manager. url: https://gitlab.unihannover.de/tci-gateway-module/device-manager/blob/master/README.md (visited on 02/14/2020)..
[36] Google and TCI. gRPC: An RPC library and framework (TCI fork with modified native library loading). url: https://gitlab.uni-hannover.de/tci-gateway-module/grpc (visited on 02/14/2020)..
[37] TCI. Gateway-Publish: Publish-System for C# SiLA2-Servers. url: https://gitlab.unihannover.de/tci-gateway-module/gateway-publish (visited on 02/14/2020)..
[38] Microsoft. Download.NET Core 3.1. url: https://dotnet.microsoft.com/download/dotnetcore/thank-you/sdk-3.1.101-linux-arm64-binaries (visited on 01/31/2020)..
[39] TCI. Installation and Operation Instructions: Gateway-Publish. url: https://gitlab.unihannover.de/tci-gateway-module/gateway-publish/blob/master/README.md (visited on 02/14/2020)..
[40] TCI. Device-Manager: Web Frontend for C# SiLA2-Servers. url: https://gitlab.uni-hannover.de/tci-gateway-module/device-manager (visited on 02/14/2020)..
[41] Hardkernel co., Ltd., User Manual ODroid C2. url: https://magazine.odroid.com/wpcontent/uploads/odroid-c2-user-manual.pdf (visited on 02/26/2020)..

3.2 SMARTLAB – INTERAKTIVES ARBEITEN IN DIGITALISIERTER LABORUMGEBUNG

Der folgende Artikel beschreibt die Struktur eines digitalisierten Labors, die im Rahmen der Entwicklung des Messeauftritts „smartLAB III“ konzipiert und auf der Labor-Fachmesse Labvolution 2019 einem breiten Publikum präsentiert wurde. In einer Kooperation zwischen verschiedenen Partnern aus Industrie und Forschung wurde ein Konzept für ein vollständig digital integriertes und automatisiert steuerbares Labor entwickelt. Abbildung 22 zeigt ein Modell des zentralen Labors innerhalb des Messeaufbaus.

Der besondere Fokus liegt dabei auf einer Vereinheitlichung der Schnittstellen zur Herstellerübergreifenden Kommunikation. Zur Anbindung aller Geräte wird der SiLA Standard eingesetzt und es kommt ein Vorläufer des in Kapitel 3.1 (Seite 22) vorgestellten Gateway-Moduls zum Einsatz. Die Ablaufsteuerung von Laborprozessen findet durch einen Prozess-Manager statt, der mit der zentralen Schnittstelle des Laborservers kommuniziert.

Einen hohen Stellenwert im smartLAB-Projekt hat die digitale Unterstützung des im Labor tätigen Menschen. Dazu wurden verschiedene Nutzerinteraktionsmöglichkeiten im Kontext der Labordigitalisierung evaluiert. So können mit Hilfe von Apps für Tablets oder SmartGlasses Aufgaben, wie das Finden von Chemikalien oder das Auszählen von Bakterienkolonien, erleichtert werden. Während des Prozessablaufs im Labor wird der Benutzer fortwährend über verschiedene Nutzerinteraktionsgeräte mit allen wichtigen Informationen versorgt und erhält Unterstützung, zum Beispiel in Form von kontextsensitiven Sicherheitshinweisen. Die dazu notwendige Infrastruktur zur Anbindung von Nutzerinteraktionsgeräten wird im Folgenden ebenfalls beschrieben.

Die in dieser Fachveröffentlichung dargestellten Prinzipen zur digitalisierten Laborinfrastruktur wurden in weiteren Arbeiten verfeinert und ergänzt und führten schließlich zu dem am Ende dieser Arbeit dargestellten Ergebnis.

ABBILDUNG 22: COMPUTERMODELL DES MESSEAUFBAUS „SMARTLAB“. DAS LABOR BESTEHT AUS VARIABEL KOMBINIERBAREN MODULEN, IN DIE TEILWEISE LABORGERÄTE DIREKT INTEGRIERT SIND.

Chemie Ingenieur Technik **Forschungsarbeit** 285

smartLAB – Interaktives Arbeiten in digitalisierter Laborumgebung

Marc Porr[1], Daniel Marquard[1], Nils Stanislawski[1], Jonas Austerjost[1], Mario Russo[2], Simon Bungers[2], Christoph Klimmt[3], Thomas Scheper[1], Sascha Beutel[1] und Patrick Lindner[1,*]

DOI: 10.1002/cite.201800090

Das smartLAB ist ein Verbund aus akademischen und nicht-akademischen Partnern mit dem Ziel eine realistische Vision des Labors der Zukunft zu entwickeln. Hierfür wird eine digitale und interaktive Laborumgebung geschaffen, die den Menschen im Laboralltag anleitet und unterstützt, nicht ersetzt. In diesem Artikel werden dabei die Gebiete Geräteansteuerung, Workflow-Entwicklung, Dokumentation und Nutzerinteraktion beleuchtet sowie das Zusammenspiel dieser Bereiche. Außerdem wird die hardwareseitige Umsetzung dargestellt und wichtige Konzepte erläutert.

Schlagwörter: Labordigitalisierung, Labor-Informations-Management-System, Mensch/Maschine-Interaktion, smartLAB

Eingegangen: 27. Juni 2018; *revidiert:* 01. Oktober 2018; *akzeptiert:* 27. November 2018

smartLAB – Working Interactively in a Digitalized Laboratory Environment

SmartLAB is a cooperation of academic and non-academic institutions with the aim to design a vision for the laboratory of the future. Therefore, a digital and interactive laboratory environment is created to guide and support people in the laboratory, but not to replace them. In this article, the areas device integration, workflow development, documentation, and user interaction as well as the interaction of these areas are discussed. Furthermore, the hardware integration is described, and important concepts are explained.

Keywords: Human-machine interactions, Laboratory digitalization, Laboratory information management system, smartLAB

1 Einleitung

Das smartLAB ist ein Verbund aus akademischen und nicht-akademischen Partnern mit dem Ziel, eine realistische Vision des Labors der Zukunft zu entwickeln. Dafür wurde ein eigenes Laboreinrichtungskonzept entwickelt, das aus sechseckigen Waben aufgebaut und modular zusammenstellbar ist, so dass das Labor an wechselnde Anforderungen angepasst werden kann. Bei einem Teil der Waben wurden Laborgeräte wie Waagen oder Magnetrührer in diese integriert, damit die Wabenoberfläche für Laborarbeiten zur Verfügung steht, wenn die entsprechenden Geräte nicht benötigt werden (Abb. 1) [1]. Außerdem werden im smartLAB neue Technologien für den Einsatz im Laborumfeld erprobt, wie z. B. die Unterstützung durch für die Mensch/Maschine-Interaktion zugelassene Robotik oder der Einsatz von 3D-Druck für die Herstellung individualisierter Materialien [2]. Einer der größten Fokuspunkte des smartLABs ist die Labordigitalisierung.

Digitalisierung im Labor wird auf unterschiedliche Weise gestaltet. So finden vermehrt Smartdevices wie Smartphones und Datenbrillen Einsatz im Labor. Es wurden in den letzten Jahren verschiedene wissenschaftliche Apps entwickelt, die das Arbeiten im Labor erleichtern sollen [3]. Die von Sim et al. publizierte GelApp ermöglicht die Dokumentation und das densitometrische Auswerten von Gelelektrophorese-Experimenten [4]. Feng et al. nutzten Datenbrillen zur Auswertung von immunchromatographischen Assays und Austerjost et al. stellten eine App vor, die Bakterienkolonien auf Agarplatten erkennt und auszählt [5, 6].

[1]Marc Porr, Dr. Daniel Marquard, Nils Stanislawski, Jonas Austerjost, Prof. Dr. Thomas Scheper, Dr. Sascha Beutel, Dr. Patrick Lindner
lindner@iftc.uni-hannover.de
Leibniz Universität Hannover, Institut für Technische Chemie, Callinstraße 5, 30167 Hannover, Deutschland.

[2]Mario Russo, Dr. Simon Bungers
Labfolder GmbH, Elsenstraße 106, 12435 Berlin, Deutschland.

[3]Prof. Dr. Christoph Klimmt
Hochschule für Musik, Theater und Medien Hannover, Institut für Journalistik und Kommunikationsforschung, Expo Plaza 12, 30539 Hannover, Deutschland.

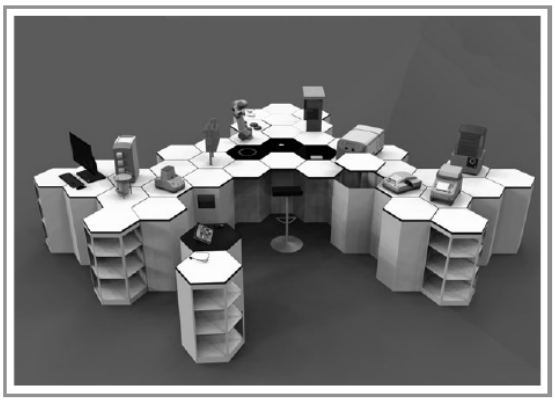


Abbildung 1. 3D-Zeichnung des smartLAB-Showlabors. In die drei schwarzen Waben in der Mitte des Labors sind (von links nach rechts) ein Magnetrührer, ein optisches Messsystem zur Bestimmung von optischer Dichte in Schüttelkolben und eine Waage integriert. Diese Integration ist so ausgeführt, dass die Oberflächen für Laborarbeit genutzt werden können, solange die Geräte nicht in Benutzung sind.

In Laboren spielt die Dokumentation eine entscheidende Rolle. Immer mehr werden hierbei die klassischen papierbasierten Laborbücher von elektronischen Laborbüchern (*electronic lab notebooks*, ELNs) ersetzt. Bereits 2011 gab es mehr als 35 unterschiedliche Anbieter von ELNs, die sich aber deutlich in ihren Anwendungsbereichen unterschieden. So waren einige nur für bestimmte Fachgebiete wie die Chemie oder Biologie geeignet und andere nur für Qualtitätssicherung oder Forschung und Entwicklung [7]. Eine Analyse über gewünschte und benötigte Funktionen eines ELN für das universitäre Life-Science-Umfeld wurde von Dirnagl et al. durchgeführt [8]. Ob die Verwendung eines ELN einen Mehrwert für die biomedizinische Forschung bietet, wurde von Guerrero et al. untersucht [9]. Hierbei zeigte sich, dass über 85 % der befragten Wissenschaftlerinnen, Wissenschaftler und Studierenden die Arbeit mit ELNs positiv bewerteten, besonders in Bezug auf die Dokumentation von Experimenten und das Teilen von Daten mit Kollegen.

Ein weiterer entscheidender Aspekt ist die digitale Ansteuerung und Vernetzung von Laborgeräten. Aktuell verwendet nahezu jeder Laborgerätehersteller sein eigenes proprietäres System, das nicht mit denen anderer Hersteller kompatibel ist. Vorhandene Geräte müssen jedoch auf die gleiche, zentrale und einfache Weise gesteuert werden können wie neu angeschaffte Geräte beliebiger Hersteller, auch wenn diese unterschiedliche Hardware- oder Softwareschnittstellen verwenden, um die Akzeptanz der Nutzer für Labordigitalisierung zu gewährleisten. Bisher konnte sich leider in der Laborumgebung noch kein allgemeiner Standard zur Gerätekommunikation etablieren. Ein Ansatz zur vollständigen digitalen Integration der Laborumgebung muss jedoch dazu in der Lage sein, prinzipiell sämtliche Laborgeräte einbinden und ansteuern zu können. Zurzeit ist die Integration von Laborgeräten verschiedener Hersteller in ein LIMS (Labor-Informations-Management-System) meist komplex und mit viel Entwicklungsaufwand verbunden. In der Literatur werden verschiedene Ansätze für solche LIMSs beschrieben, allerdings haben diese einen starken Fokus auf hoch automatisierte Prozesse [10, 11]. Ein Konzept für die allgemeine Einbindung von Laborgeräten findet sich bei Lütjohann et al. [12].

Im smartLAB wird versucht, die verschiedenen Digitalisierungsaspekte zusammenzubringen und so eine interaktive Laborumgebung zu schaffen. Dafür muss das digitale Labornetzwerk im smartLAB dazu in der Lage sein, sämtliche Laborgeräte zu steuern, Daten zentral zu verwalten und Protokollschritte auf Geräten auszuführen. Die Bedienung, Parametrisierung und Programmierung von einzelnen Geräten wird so im Idealfall obsolet und der Anwender im Labor kann sich während der Durchführung eines Experiments ganz auf das Probenhandling konzentrieren. Das Labor unterstützt und führt den Mitarbeiter durch die einzelnen Arbeitsschritte. Hierfür wird eine intuitive Visualisierung benötigt, z. B. in Form von Piktogrammen, die von einer Datenbrille angezeigt werden. Im Folgenden wird das Digitalisierungskonzept des smartLABs erläutert und die hardwareseitige Umsetzung dargestellt.

2 Infrastruktur

Mit SiLA (*standard in laboratory automation*) existiert bereits ein Standard, der Laborgeräte mit ihren Parametern, Funktionen, Zuständen sowie möglichen (Bedienungs-)Fehlern abbilden kann und eine darauf basierende Ansteuerung ermöglicht [13]. SiLA wird im smartLAB als einheitliches Protokoll zur Anbindung aller Geräte eingesetzt, so dass die Steuersoftware des Labors unabhängig von den verwendeten Geräten wird. So kann bspw. eine Waage eines Herstellers problemlos gegen ein vergleichbares Modell eines anderen ausgetauscht werden [13]. Die Softwarebefehle zur Ansteuerung ändern sich dadurch nicht, da beide durch den SiLA-Standard kommunizieren. SiLA ist als TCP/IP-Protokoll entwickelt und nutzt Ethernet oder WLAN zur Datenübertragung. Ethernet-Netzwerke verfügen bei der Geräteanbindung über weitreichende Vorteile: sie bieten schnelle Übertragungsraten, sind unempfindlich gegen Störungen und es können große Distanzen mit Ethernet-Kabeln überbrückt werden. Ferner ist Ethernet seit Jahrzehnten Standard für Büro- und Heimnetzwerke, so dass vielfach eine bestehende Infrastruktur genutzt werden kann.

Viele moderne Geräte auf dem Consumer-Markt bieten heute drahtlose Schnittstellen an. Es werden verschiedene solcher Geräte eingesetzt, um die Darstellung des laufenden Prozesses im smartLAB für den Benutzer zu realisieren. Drahtlose Kommunikation wird dazu jedoch nur verwendet, wenn eine kabelgebundene Lösung nicht realisierbar ist, z. B. bei Handheld-Geräten. Neben der geringeren Geschwindigkeit von Wireless-LAN-Netzwerken spielen hier Sicherheits- und Zuverlässigkeitsaspekte eine entscheidende

Rolle. Gerade in Umgebungen, in denen sich viele Funknetzwerke überlagern, ist die Verbindungsqualität oft schlecht und die Kommunikation unzuverlässig. Dazu kommt bei Bluetooth-Verbindungen noch eine geringe Reichweite und die Tatsache, dass SiLA als TCP/IP-Protokoll nicht direkt kompatibel zu Bluetooth ist.

Die digitale Infrastruktur des smartLABs nutzt einen Virtualisierungsserver (Abb. 2) als Basis. Dieser simuliert mehrere eigenständige Computer mit ihrer Hard- und Software, sog. virtuelle Maschinen (VMs). Auf diese Weise können die benötigten Programme zur Anbindung von Geräten voneinander isoliert werden und so bspw. auf dem Betriebssystem ausgeführt werden, das am besten geeignet ist. Ferner bietet *virtual computing* die Möglichkeit, den momentanen Zustand einer VM zu speichern, später wiederherzustellen oder zu duplizieren. Dadurch wird die Datensicherheit erhöht und der Administrationsaufwand sinkt. Da die Ressourcen des Laborservers je nach Bedarf dynamisch auf die einzelnen VMs verteilt werden, ist die Lastverteilung besser und es muss weniger redundante Hardware angeschafft werden als bei physikalisch eigenständigen Systemen [14].

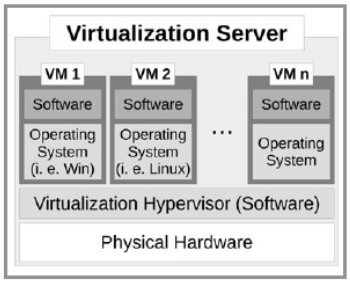


Abbildung 2. Schematische Darstellung der Funktionsweise eines Virtualisierungsservers. Die physikalische Hardware simuliert mithilfe einer speziellen Software (Hypervisor) mehrere verschiedene, unabhängige Hardware-Systeme. Auf diesen läuft je ein Betriebssystem, das wiederum die benötigte Software ausführt. Für die Betriebssysteme und Software in den virtuellen Maschinen besteht kein Unterschied zwischen physikalisch vorhandener oder simulierter Hardware.

Durch den Einsatz virtueller Maschinen ist die Infrastruktur so flexibel, dass in einem nächsten Schritt ebenfalls die Implementierung der LIMS-Funktionalität als Cloud-Lösung möglich wäre. Zurzeit scheint jedoch ein lokaler Server geeigneter, um während der Entwicklungszeit nicht von einer schnellen Internetverbindung abhängig zu sein und um eine einfache Wartbarkeit zu gewährleisten.

2.1 Anbindung von Geräten ohne Ethernet-Schnittstelle durch die ComBox

Mittels des SiLA-Standards wird die gesamte Gerätekommunikation im smartLAB mit TCP/IP-Verbindungen über Ethernet-Schnittstellen durchgeführt. Da auf absehbare Zeit jedoch nicht alle angebotenen Geräte über eine solche verfügen und ferner die Anbindung von Bestandsgeräten sichergestellt werden muss, wird im smartLAB die Anbindung nicht-ethernetfähiger Geräte mithilfe eines Gateway-Moduls (ComBox) realisiert, das zwischen Gerät und Laborserver geschaltet wird, so dass solche Geräte trotzdem via Ethernet und SiLA-Interface integriert werden können (Abb. 3). Somit wird es für die Steuersoftware und damit letztendlich den Benutzer irrelevant, ob ein Gerät direkt über Ethernet oder über eine serielle Schnittstelle via ComBox an das Labornetzwerk angeschlossen ist.

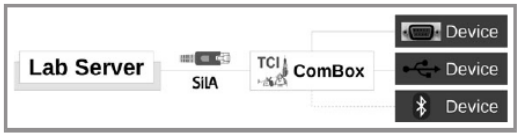


Abbildung 3. Darstellung der Anbindung von Nicht-Ethernetgeräten mithilfe eines Gateway-Moduls (ComBox). Die im Laboraufbau untergebrachte ComBox wird über ein Ethernet-Kabel mit dem zentralen Laborserver verbunden. Die Kommunikation mit dem Server findet mithilfe des TCP/IP-basierten SiLA-Standards statt. An die ComBox wird ein Laborgerät angeschlossen, das selbst über keine Ethernet-Schnittstelle verfügt. So werden z. B. seriell, USB- oder drahtlos (bspw. über Bluetooth) angebundene Geräte angeschlossen. Die ComBox übernimmt die Übersetzung der SiLA-Befehle in die entsprechende vom Gerät verlangte Form mithilfe eines entsprechend entwickelten Gerätetreibers.

Die ComBox ist ein Einplatinencomputer, der im Laboraufbau nah bei dem anzubindenden Gerät untergebracht wird. Dadurch werden die Kabellängen von Nicht-Ethernetverbindungen oder die Funkstrecken (z. B. bei Bluetooth-Kommunikation) kurzgehalten. Jedes Nicht-Ethernetgerät wird mit einer ComBox angebunden, auf der eine speziell entwickelte Treiberanwendung ausgeführt wird, die zwischen SiLA und dem Geräteprotokoll übersetzt. Mit dieser Methode können viele Laborgeräte, die über serielle, z. B. RS-232, oder USB-Schnittstellen verfügen, angebunden werden. Die Nachteile dieser Verbindungsmethoden, wie fehlende Standardisierung oder Probleme bei langen Anschlussleitungen, werden so vermieden.

3 smartLAB-Netzwerk

Das smartLAB-Netzwerk besteht aus drei Ebenen und basiert auf einer Struktur mit einer Server-Client-Logik (Abb. 4). So wird sichergestellt, dass Geräte auf einheitliche Art ferngesteuert werden können und die Prozessintelligenz an zentraler Stelle gesammelt vorliegt. Dies ist für eine intuitive Nutzerführung ebenso vorteilhaft wie für eine mögliche Zertifizierbarkeit der Laborprozesse.

Auf der untersten Ebene stehen die einzelnen Laborgeräte. Diese stellen ihre Funktionalität mit einer standardi-

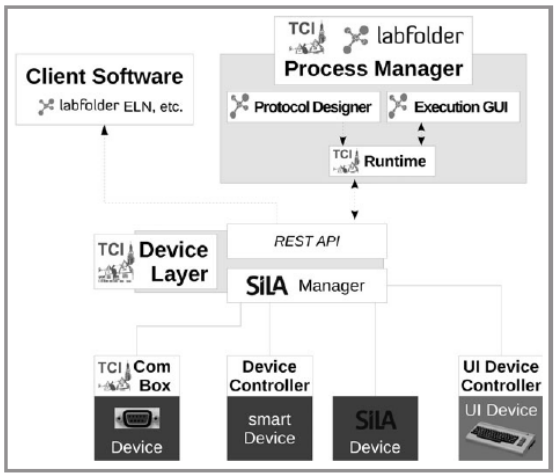


Abbildung 4. Übersicht der Netzwerkstruktur und Kommunikation zwischen den Softwarekomponenten im smartLAB. Gestrichelte Linien repräsentieren Datenströme, während durchgezogene Linien physikalische (Kabel-)Verbindungen darstellen. Die Geräte werden mit dem Device Layer über das SiLA-Protokoll verbunden, der über eine REST-API die Gerätefunktionalität vereinheitlicht zur Verfügung stellt. Im Process Manager werden Laborprotokolle angelegt und ausgeführt. Weitere Software zur Überwachung oder Protokollierung des Laborprozesses kann ebenfalls über die API angebunden werden.

sierten Schnittstelle zur Verfügung, ohne die Zusammenhänge in einem Laborprozess kennen zu müssen. Je nach Hardwareschnittstelle und verwendetem Kommunikationsprotokoll sind verschiedene Maßnahmen nötig, um ein Gerät ans smartLAB anzubinden. Eine Sonderstellung nehmen dabei Geräte zur Nutzerinteraktion ein. Dabei handelt es sich nicht um Laborgeräte, sondern um verschiedene mobile Endgeräte. Im smartLAB werden zu diesem Zweck Datenbrillen, Touch-Oberflächen und Sprachsteuerungshardware eingesetzt.

Die mittlere Ebene bildet der *TCI Device Layer* (TCIdl), dieser regelt alle Geräteanfragen, stellt die Verfügbarkeit der Geräte sicher und koordiniert die Gerätenutzung. Die Kommunikation zwischen Laborgeräten und TCIdl erfolgt hierbei über eine SiLA-Schnittstelle. Für die Kommunikation mit übergeordneter Clientsoftware wird eine REST-API (*representational state transfer-application programming interface*) angeboten. REST ist ein Standard für Webprogrammierschnittstellen (Web-APIs), die sicherstellt, dass alle Teilnehmer die gleichen Befehle und Antworten erwarten. Die oberste Ebene wird durch Clientsoftware und Nutzeranwendungen eingenommen. Auf dieser Ebene befindet sich die eigentliche Prozessintelligenz. Die Prozesssteuerung und Planung wird im *Process Manager* durchgeführt. Auch Software zur strukturierten Datenspeicherung (ELN) oder Protokollierung (Audit-Trail), kann als Client über die REST-API in das smartLAB-Netzwerk eingebunden werden.

3.1 Zentralisierte Steuerung von Laborgeräten durch den TCI Device Layer

Der TCIdl ist die einheitliche Laborserveranwendung und der Knotenpunkt des leistungsfähigen LIMS des smartLABs. Er enthält ebenfalls eine Datenbank, in der sämtliche Parameter, Gerätedaten, Messwerte und mögliche Gerätefehler eines Prozesses abgelegt werden. Das LIMS des smartLABs verfügt über eine Abstraktionsebene zur Gerätekommunikation, um die zentrale Steuerung aller Laborgeräte sowie die Verfügbarkeit aller gesammelten Daten sicherzustellen. Der TCIdl regelt die Anbindung und Kommunikation aller Geräte und bietet eine zentrale Programmschnittstelle. Über diese können verschiedenste Softwarekomponenten auf die Laborinfrastruktur zugreifen und Messdaten und Prozessparameter abfragen sowie Steuerbefehle an Laborgeräte senden.

So kann bspw. ein ELN ein Experiment live aufzeichnen oder eine Audit-Trail-Software eine Analyse lückenlos und ohne Zutun des Laborarbeiters dokumentieren. Übertragungsfehler oder Ungenauigkeiten in der Dokumentation bleiben aus. Abb. 5 zeigt schematisch die Anbindung von Laborgeräten an den TCIdl. Die Gerätekommunikation findet mit SiLA statt. Geräte, die vom Hersteller bereits SiLA-konform ausgeliefert werden, können direkt mit dem TCIdl kommunizieren. Geräte, die eine proprietäre Steuersoftware durch den Hersteller mitgeliefert bekommen, müssen über einen Device-Controller angebunden werden, der die SiLA-Befehle entsprechend in die API des Herstellers übersetzt. Für Geräte ohne Ethernet-Schnittstelle, und folglich ohne eigene SiLA-Unterstützung, bietet die ComBox durch einen implementierten Device-Controller dem TCIdl eine SiLA-konforme Schnittstelle. Geräte zur Nutzerinteraktion werden ebenfalls durch einen Device-Controller mit dem SiLA-Protokoll angesteuert. Diese Geräte verfügen softwareseitig jedoch über einen anderen Befehlssatz, der die Darstellung von Prozessschritten sowie die Prozesssteuerung durch Nutzerbefehle erlaubt.

Der TCIdl selbst bietet nach außen zwei Schnittstellen an. Einerseits den SiLA-Manager, an den alle Geräte angebunden werden. Andererseits eine Webschnittstelle in Form einer REST-API, über die Software das Labor steuern bzw. Daten abfragen kann. Intern werden die Zustände der Geräte in Form „virtueller Geräteschatten“ (*device shadows*) repräsentiert. Diese stellen eine digitale Kopie des Ist-Zustands des Geräts dar, der durch den SiLA-Manager mit dem realen Zustand des Geräts synchron gehalten wird. Außerdem ist über eine angebundene Datenbank die gesamte Historie von Messergebnissen, Analysenläufen und dazu verwendeten Parameterwerten Teil der digitalen Gerätepräsentation. Der Verlauf eines Messprozesses wird vom Einschalten und Initialisieren des Geräts über den ganzen Verlauf von Messungen (mit allen eingestellten Parametern) bis zum Abschalten inkl. möglicher Fehler und deren Behebung in der Datenbank hinterlegt.

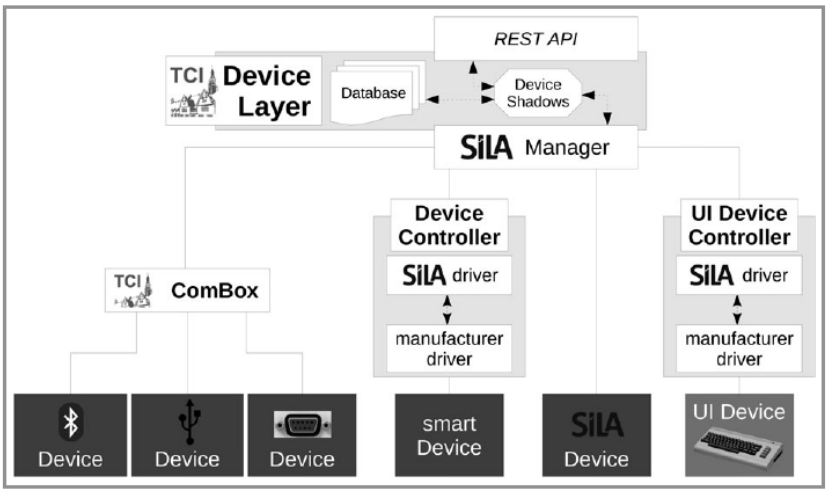


Abbildung 5. Zentralisierte Geräteanbindung im smartLAB durch den Device Layer mit dem SiLA-Protokoll.

Über die API-Schnittstelle kann der TCIdl nach den angeschlossenen Geräten im Labor gefragt werden. Zu jedem Gerät kann der Ist-Zustand genauso abgerufen werden, wie die Historie. Für diese Art von Anfragen ist ausschließlich lesender Zugriff auf die API-Schnittstelle notwendig. Dieser wird uneingeschränkt allen Clients im Labornetzwerk gewährt. Für die Steuerung der Geräte zur Ausführung eines Prozesses muss eine Software entsprechende Rechte zum schreibenden Zugriff auf das Labor haben. Über einen Identifikationsschlüssel kann sich die *Runtime* des Process Managers am TCIdl identifizieren, so dass nur ihr Änderungen an den virtuellen Geräteschatten erlaubt sind. Dabei kann sich nur eine einzige Runtime verbinden, so dass die Prozesssteuerung exklusiv an zentraler Stelle stattfindet. Pro Labor können jedoch mehrere parallel ausgeführte Instanzen des TCIdl existieren, die sich Geräte teilen. Wollen mehrere Mitarbeiter gleichzeitig im Labor arbeiten, startet jeder auf „seinem" TCIdl ein Protokoll. Soll ein Prozessschritt auf einem Gerät gestartet werden, das zurzeit bereits in Verwendung ist, wird die Ausführung so lange pausiert, bis das entsprechende Gerät vom anderen Protokoll wieder freigegeben wurde.

4 Nutzerinterface und Prozesssteuerung

Für eine flexible und einfache Arbeit im Labor muss ein intuitiv bedienbares Nutzerinterface (Frontend) geschaffen werden. Dies betrifft sowohl das Designen und Parametrisieren von Experimenten, wie auch die Nutzerführung während des Ablaufs. Wird ein Prozess durchgeführt, müssen die entsprechenden Gerätebefehle und -anfragen an den TCIdl gesendet und die Geräteantworten und Messwerte interpretiert werden. Die logische Verkettung von einzelnen gerätegebundenen Prozessschritten ergibt ein Ablaufprotokoll. Die Gerätesteuerung in Form des TCIdl muss den Zusammenhang zwischen einzelnen Schritten und den assoziierten Geräten nicht kennen. Die gesamte Prozessintelligenz liegt zentral im Process Manager vor. Dies erleichtert die Prozessplanung, Administration und auch die eventuell später gewünschte Zertifizierung von Abläufen. Der Process Manager hat mehrere Bestandteile. Der *Protocol Designer* ist eine Bedienoberfläche zur Erstellung von Laborprotokollen. Fertige Protokolle werden durch die Runtime gelesen und ausgeführt. Während der Ausführung erhält der Nutzer eine direkte visuelle Rückmeldung zum Status des Protokolls durch das *Execution GUI*.

4.1 Prozessplanung und Durchführung

Der Process Manager fasst die Frontend-Software im smartLAB an einer zentralen Stelle zusammen. Auf diese Weise kann der Mitarbeiter Prozessdesign und -ausführung in einer intuitiven und logischen Maske durchführen. Der Process Manager beinhaltet den Protocol Designer zur graphischen Prozessplanung, das Execution GUI zur Visualisierung eines laufenden Prozesses sowie das *Debug GUI* zur technischen Überwachung und die Runtime. Diese übersetzt Nutzerbefehle und Protokolle in Geräteanfragen an den Device Layer (Abb. 6).

Der Protocol Designer stellt alle Laborgeräte mit ihren Funktionen und Parametern graphisch aufbereitet dar und erlaubt die einfache Erstellung von Experimenten und Protokollen per Drag-and-drop. Dazu fragt er über die REST-API-Schnittstelle des TCIdl Informationen über die Infrastruktur im Labor an, so dass der Nutzer nur die für ihn relevanten Geräte angezeigt bekommt. Abb. 7 zeigt beispielhaft die Nutzeroberfläche bei der Erstellung eines Protokolls. Ein fertig geplanter Laborprozess wird in Form einer Protokolldatei gespeichert. Diese wird zur Durchführung von der Runtime gelesen und interpretiert. Sie kommuniziert dabei mit dem Execution GUI, das dem Nutzer den aktuellen Schritt im Kontext des Prozessablaufs dar-

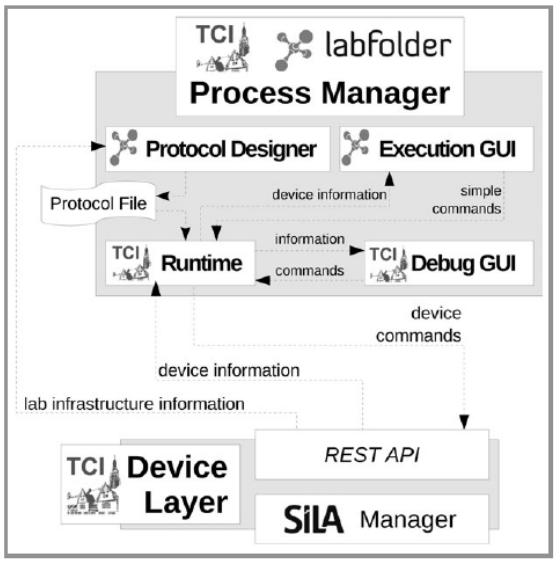


Abbildung 6. Kommunikation der Komponenten des Process Managers mit dem Device Layer. Die komplette Prozesssteuerung und Überwachung findet über die einheitliche REST-API des Device Layer statt.

stellt und ihm erlaubt steuernd oder korrigierend einzuwirken. Dabei wird er nur mit den für seine Arbeit wichtigen Daten und Informationen versorgt. Abb. 8 zeigt beispielhaft die Visualisierung eines laufenden Prozesses im Execution GUI.

Die Protokolldatei besteht aus Einzelschritten, die z. B. beschreiben, welches Gerät mit welchen Parametern eingesetzt werden soll. Die Runtime fragt zuerst beim TCIdl an, ob das als nächstes zu verwendende Gerät zur Verfügung steht. Falls dieses einen Fehler meldet, wird der Nutzer aufgefordert diesen zu beheben bevor der Ablauf fortgesetzt werden kann. Funktioniert das Gerät einwandfrei, wird es durch REST-API-Befehle an den TCIdl vollautomatisch programmiert, parametrisiert und – sobald der Mitarbeiter das Probenhandling abgeschlossen hat – gestartet.

Zu jedem Schritt in der Protokolldatei sind ebenfalls mögliche Überleitungen zu Folgeschritten gespeichert. So kann z. B. je nach Ergebnis einer Messung ein anderer Schritt folgen. Auch können die Auslöser für eine Überleitung verschieden sein. Der einfachste Fall ist das direkte Fortfahren mit dem Folgeschritt. Dies setzt nicht voraus, dass die auf einem Gerät gestartete Methode zu diesem Zeitpunkt bereits abgeschlossen ist. So ist ein asynchrones Arbeiten möglich, bei dem der Nutzer andere Arbeiten tätigt, während bspw. eine Zentrifuge läuft.

Eine andere Variante eines Prozessschrittübergangs ist das Abwarten des Abschlusses einer Methode auf einem Gerät. Soll z. B. eine Wägeoperation durchgeführt werden, so ist dieser Schritt so lange aktiv, bis der Mitarbeiter die geforderte Menge des Stoffes in der gewünschten Toleranz auf dem Wägeschälchen platziert hat. Bei einigen Schritten, z. B. beim Pipettieren, ist ein manuelles Quittieren des Abschlusses durch den Nutzer notwendig. Dies kann durch das Klicken eines Buttons in dem Execution GUI oder durch das Auslösen einer „Nächster-Schritt-Funktion“ auf einem alternativen UI-Gerät erfolgen. Ist z. B. ein Gerät zur Sprach-

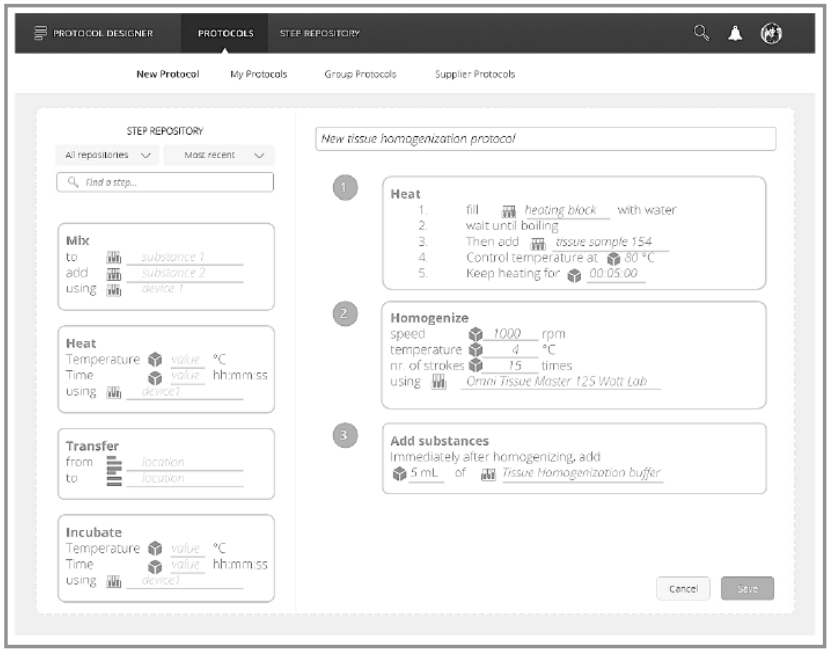


Abbildung 7. Nutzeroberfläche des Protocol Designers zur Prozessplanung.

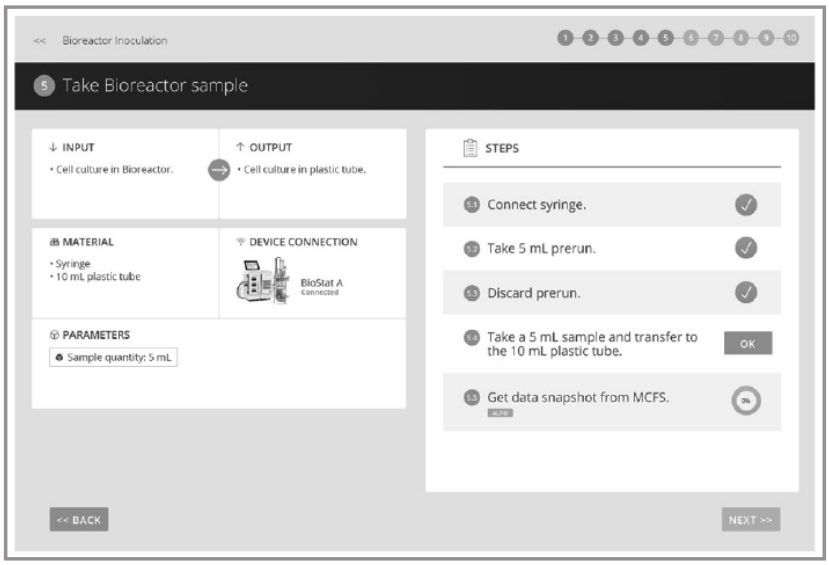


Abbildung 8. Nutzeroberfläche während der Prozessdurchführung im smartLAB.

steuerung im Labornetzwerk vorhanden, so kann der Nutzer den Schritt auch durch den Sprachbefehl „next" abschließen.

Es müssen auch Prozessschritte, die keine Geräteinteraktion erfordern, abgebildet werden können, damit die Prozessführung entsprechend gestaltet und die Abläufe dokumentiert werden können. Ist bspw. in einem mikrobiologischen Labor das Ausplattieren einer Probe notwendig, muss das Labor so lange auf den Nutzer warten, bis dieser Prozessschritt abgeschlossen ist. Sollen Daten zu diesem Schritt, z. B. ein Foto der Platte, aufgezeichnet werden, so müssen diese ebenfalls direkt an das ELN zur Protokollierung weitergeleitet werden.

Zusätzlich zum für den Endnutzer gedachten Execution GUI kann sich der Laborleiter oder Administrator den detaillierten Ablauf der Gerätebefehle und -parameter in einem Debug GUI anzeigen lassen. Hier werden die API-Anfragen und -Antworten protokolliert und der Hardware-Status der Geräte kann überwacht werden. Dieses Werkzeug stellt auch für den Gerätetechniker wertvolle Informationen zu Funktion und Zustand der Geräte bereit und erlaubt ihm, steuernd in die Gerätekommunikation einzugreifen.

4.2 Geräte zur Nutzerinteraktion

Die digitale Revolution der letzten Jahre hat im Verbraucherbereich viele neue Formen der Nutzerinteraktion verfügbar gemacht. Einige dieser Techniken lassen sich für das Labor adaptieren und bieten, unter der Voraussetzung einer einfachen und lückenlosen Integration, viele Vorteile für den Mitarbeiter. So erlaubt eine Steuerung von Laborgeräten über Sprachbefehle das Bedienen einfacher Funktionen ohne dabei die Hände frei haben zu müssen [15].

Zur Unterstützung des Laborarbeiters bei neuen oder komplizierten Tätigkeiten werden verschiedene Geräte in die Infrastruktur integriert. Mit einer Datenbrille kann ihm beim Pipettieren komplexer Assays die Reihenfolge der Schritte direkt in seinem Sichtfeld eingeblendet werden. Oder er wird bei der Handhabung gefährlicher Stoffe durch das sofortige Bereitstellen von Sicherheitsinformationen unterstützt. Hierbei ist entscheidend, dass die Runtime die verfügbaren Informationen filtert und kontextsensitiv nur die Daten der Brille zur Anzeige weiterleitet, die dem Nutzer in der aktuellen Situation helfen, ohne ihn abzulenken. Beispielhaft ist in Abb. 9 die Anzeige auf der Datenbrille bei der Probennahme aus einem Bioreaktor gezeigt. Hierbei ist auch zu beachten, dass eine längere ununterbrochene Nutzung eines aktuellen Datenbrillenmodells im Arbeitsalltag negative Folgen aufgrund der zusätzlichen Belastung haben kann [16].

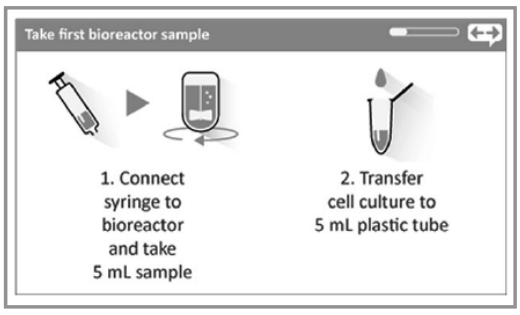


Abbildung 9. Anzeige für den Labormitarbeiter auf der Datenbrille bei der Probennahme aus einem Bioreaktor. Der Prozessschritt wird mithilfe intuitiv verständlicher Piktogramme stark vereinfacht dargestellt, um den Mitarbeiter nicht abzulenken, sondern ihm eine praktische Hilfestellung bei schwierigen oder selten durchgeführten Arbeiten zu geben.

Der Einsatz der Datenbrille ist ebenfalls zeitlich begrenzt zur Unterstützung bei monotonen oder anstrengenden Tätigkeiten denkbar. Sie kann über eine Kamera direkt Fotos des Sichtbereichs aufnehmen, die dann z. B. durch einen Bildverarbeitungsalgorithmus automatisiert ausgewertet werden können [4]. So könnte ein farbenblinder Mitarbeiter eine Hilfestellung bei der Beurteilung einer Indikatorreaktion mit Farbumschlag erhalten. Oder eine zeitaufwendige Aufgabe, wie das Auszählen von Bakterienkolonien in Petrischalen, kann dem Nutzer abgenommen werden. Dazu wird ein Foto der Petrischale aufgenommen und durch einen Bildverarbeitungsalgorithmus automatisiert ausgewertet [6]. Die Ergebnisse werden direkt im ELN gespeichert, so dass die zeitraubende manuelle Auszählung ebenso wie die händische Dokumentation nicht mehr nötig sind. Eine Anwendung, die den Mitarbeiter durch Prozessieren der Bilddaten des aktuellen Sichtbereichs bei der Identifikation gesuchter Gebinde in einem Chemikalienlager unterstützt, befindet sich ebenfalls in der Erprobung.

Die im smartLAB verwendete Datenbrille bringt bereits eine rudimentäre Sprachsteuerung mit, die einfache Befehle wie die Anweisung next verarbeiten kann. Für eine präzisere Spracherkennung mit kontextsensitiver Erkennung von Zielobjekten und Anweisungen sind jedoch weit komplexere *Natural-Language-Processing*-Algorithmen nötig. Diese benötigen viel Rechenleistung und werden deshalb auf externen Servern ausgeführt. Verschiedene Hersteller bieten solche Lösungen für den Verbrauchermarkt an. Solche Geräte lassen sich ebenfalls gut für die Unterstützung der Laborarbeit adaptieren und können in das smartLAB-Netzwerk integriert werden [15]. Hierbei muss jedoch abgewogen werden, ob die Weiterleitung eines Teils der im Labor erzeugten Daten an Server externer Unternehmen vertretbar und erwünscht ist.

In den letzten Jahren haben berührungssensitive Bildschirme die Interaktion mit Geräten von Grund auf verändert. Durch Touchscreens werden Smartphones und Tablets intuitiv bedienbar und die Software kann sich den Bedürfnissen der Nutzer viel besser anpassen. Eine solche Bedienung ist im Laborbereich jedoch schwierig zu realisieren, da tragbare Geräte dann durch die Bedienung mit Handschuhen in Kontakt mit Chemikalien, biogefährlichen Stoffen oder Mikroorganismen kommen würden. Eine denkbare Alternative ist das Bereitstellen stationärer Bedienpanels (ähnlich solchen, die in Laborgeräte integriert sind), um sicherzustellen, dass kontaminierte Hardware das Labor nicht verlassen kann. Diese sind jedoch unflexibel in der Positionierung und schwer sauber zu halten. Eine bessere Alternative stellt das Funktionalisieren von Labortisch- oder Wandoberflächen dar. Dazu werden sogenannte Touch-Beamer eingesetzt, die ein Bild auf eine beliebige Oberfläche projizieren und die Handbewegungen des Benutzers mithilfe eines unsichtbaren Infrarotrasters erfassen. Im smartLAB wird die Bereitstellung verschiedener visueller Elemente mit dieser Technologie derzeit erprobt.

5 Zusammenfassung

Im smartLAB wird für die Geräteansteuerung der SiLA-Standard verwendet, der auf TCP/IP via Ethernet basiert. Geräte, die nicht über eine solche Schnittstelle verfügen, werden mittels ComBox in das System integriert und können so ebenfalls über den SiLA-Standard angesteuert werden.

Das zentrale Element in der Netzwerkstruktur bildet der TCI Device Layer. Dieser enthält die virtuellen Abbilder aller im Labor vorhandenen Geräte, die den aktuellen Zustand des dazugehörigen Gerätes abbilden. Außerdem wird jeder ablaufende Schritt in einer Datenbank dokumentiert. Diese Informationen bietet der TCIdl über eine REST-API an, so dass über diese Server/Client-basierte Struktur andere Software-Komponenten, wie z. B. das ELN, auf die Daten zugreifen und diese weiterverarbeiten können.

Der Process Manager ist eine Software, die eine graphische Nutzeroberfläche anbietet und aus drei Untereinheiten besteht. Der Protocol Desginer ermöglicht das Planen von Laborabläufen. In der Runtime werden die zuvor erstellten Abläufe ausgeführt. Sie leitet die benötigten Gerätebefehle an den TCIdl weiter. Im Execution GUI wird der aktuell ablaufende Prozess dargestellt und die Möglichkeit gegeben, in diesen steuernd einzugreifen. Zusätzlich zum Exection GUI werden noch weitere Visualisierungssysteme eingesetzt, z. B. Datenbrillen. Diese sollen dem Mitarbeiter im Labor wichtige Information intuitiv zur Verfügung stellen und ihn so durch den Prozess führen.

Ob das hier vorgestellte Digitalisierungskonzept für den Laboralltag geeignet ist, muss in der Praxis erprobt werden. Dafür ist es wichtig, dass geeignete Einstiegskonzepte entwickelt werden, die den Mitarbeiter im Labor abholen und verdeutlichen, welche Mehrwerte ihm ein interaktives Laborumfeld bietet.

Wir bedanken uns für die Finanzierung der Forschungsarbeiten im Rahmen einer gemeinsamen Förderung durch die Niedersächsischen Ministerien für Wissenschaft und Kultur sowie Wirtschaft, Arbeit und Verkehr.

Literatur

[1] S. Beutel, P. Lindner, C. Endres, *Nachr. Chem.* **2016**, *64 (4)*, 428 – 430. DOI: https://doi.org/10.1002/nadc.20164046793
[2] L. Raddatz, J. Austerjost, S. Beutel, *Chem. Unserer Zeit* **2018**, *52 (1)*, 42 – 50. DOI: https://doi.org/10.1002/ciuz.201700802
[3] S. K.-E. Gan, J.-K. Poon, *Sci. Phone Apps Mobile Devices* **2016**, *2 (1)*, 6. DOI: https://doi.org/10.1186/s41070-016-0009-2
[4] J.-Z. Sim, P.-V. Nguyen, H.-K. Lee, S. K. Gan, *Nat. Methods Appl. Notes* **2015**, *1 – 2*. DOI: https://doi.org/10.1038/an964

[5] S. Feng, R. Caire, B. Cortazar, M. Turan, A. Wong, A. Ozcan, *ACS Nano* **2014**, *8 (3)*, 3069 – 3079. DOI: https://doi.org/10.1021/nn500614k
[6] J. Austerjost, D. Marquard, L. Raddatz, D. Geier, T. Becker, T. Scheper, P. Lindner, S. Beutel, *Eng. Life Sci.* **2017**, *17 (8)*, 959 – 966. DOI: https://doi.org/10.1002/elsc.201700056
[7] M. Rubacha, A. K. Rattan, S. C. Hosselet, *J. Lab. Autom.* **2011**, *16 (1)*, 90 – 98. DOI: https://doi.org/10.1016/j.jala.2009.01.002
[8] U. Dirnagl, I. Przesdzing, *F1000Research* **2016**, *5*, 2. DOI: https://doi.org/10.12688/f1000research.7628.1
[9] S. Guerrero et al., *PLoS One* **2016**, *11 (8)*, e0160428. DOI: https://doi.org/10.1371/journal.pone.0160428
[10] I. Schmid, J. Aschoff, *Eng. Life Sci.* **2017**, *17 (11)*, 1159 – 1165. DOI: https://doi.org/10.1002/elsc.201600008
[11] T. Wu, Y. Zhou, *J. Lab. Autom.* **2014**, *19 (4)*, 381 – 393. DOI: https://doi.org/10.1177/2211068213499756
[12] D. S. Lütjohann, N. Jung, S. Bräse, *Chemom. Intell. Lab. Syst.* **2015**, *144*, 100 – 107. DOI: https://doi.org/10.1016/j.chemolab.2015.04.002
[13] H. Bär, R. Hochstrasser, B. Papenfuß, *J. Lab. Autom.* **2012**, *17 (2)*, 86 – 95. DOI: https://doi.org/10.1177/2211068211424550
[14] J. Daniels, *Crossroads* **2009**, *16 (1)*, 8 – 12. DOI: https://doi.org/10.1145/1618588.1618592
[15] J. Austerjost, M. Porr, N. Riedel, D. Geier, T. Becker, T. Scheper, D. Marquard, P. Lindner, S. Beutel, *SLAS Technol.* **2018**, *23 (5)*, 476 – 482. DOI: https://doi.org/10.1177/2472630318788040
[16] *Head-Mounted Displays – Arbeitshilfen der Zukunft*, Bundesanstalt für Arbeitsschutz und Arbeitsmedizin (BAuA), Dortmund **2016**. DOI: https://doi.org/10.21934/baua:praxis20160809

3.3 VORSTELLUNG EINES VIRTUELLEN SPRACHASSISTENTEN FÜR DIE INTUITIVE KONTROLLE VON LABORGERÄTEN

Zur Interaktion des digitalisierten Labors mit dem Benutzer bietet sich der Einsatz verschiedener Nutzerinteraktionsmedien an. Im Heimanwenderbereich werden viele digitale Werkzeuge bereits seit längerem eingesetzt und haben bewiesen, dass sie unter bestimmten Bedingungen die Steuerung digitaler Systeme erheblich vereinfachen können. Der folgende Artikel beschäftigt sich mit der Adaption von *Smart Speakern* für die Anwendung im Laborbereich. Diese Geräte zeichnen sich durch die Anbindung an leistungsfähige Server mit mitlernenden Algorithmen zur Verarbeitung natürlicher Sprache aus. Dadurch ist eine intuitive und freihändige Bedienung möglich.

Beispielhaft wird der Aufbau eines Systems auf Basis des *Echo-SmartSpeakers* des Herstellers *Amazon Inc. (Seattle, Washington, USA)* gezeigt. Um eine Steuerung von Laborgeräten zu ermöglichen, müssen die Funktionen dieser Geräte zuerst digital zur Verfügung gestellt werden. Dieser digitale Geräteschatten kann dann mit Hilfe einer als *Cloud-Computing-Service* gehosteten Web-Anwendung kommunizieren, so dass aus der Cloud heraus Aktionen auf dem Laborgerät ausgelöst und Daten ausgelesen werden können. Zur Vermittlung zwischen dem vom Hersteller des *Smart Speakers* zur Verfügung gestellten Sprachverarbeitungssystem und dem Geräteservice wird ein so genannter „*Lambda-Service*“ benötigt. Dieser bindet bestimmte Sprachbefehle an bestimmte Geräteaktionen. Abbildung 23 zeigt schematisch den Datenstrom vom Sprachbefehl des Benutzers bis hin zur Geräteaktion.

Zur praktischen Verdeutlichung der Vorgehensweise werden im folgenden Artikel eine Waage und ein Spektralphotometer zur Steuerung mit einem *Smart Speaker* angebunden. Der Verwender kann freihändig mit diesen Geräten interagieren und sie so besser in seinen Arbeitsablauf integrieren. Das gezeigte Konzept kann unabhängig von den verwendeten Geräten adaptiert und zur Steuerung beliebiger technischer Systeme eingesetzt werden.

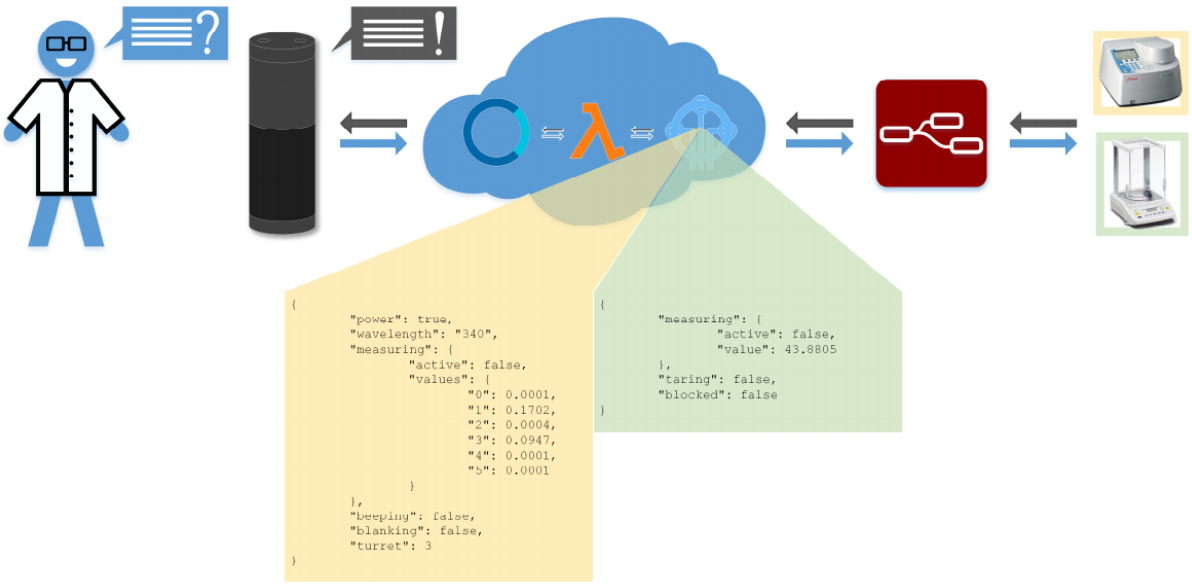


ABBILDUNG 23: GRAPHICAL ABSTRACT DES ARTIKELS "INTRODUCING A VIRTUAL ASSISTANT TO THE LAB: A VOICE USER INTERFACE FOR THE INTUITIVE CONTROL OF LABORATORY INSTRUMENTS".

Original Research

SLAS Technology
2018, Vol. 23(5) 476–482

DOI: 10.1177/2472630318788040
journals.sagepub.com/home/jla
SAGE

Introducing a Virtual Assistant to the Lab: A Voice User Interface for the Intuitive Control of Laboratory Instruments

Jonas Austerjost[1,2], Marc Porr[1], Noah Riedel[1], Dominik Geier[2], Thomas Becker[2], Thomas Scheper[1], Daniel Marquard[1], Patrick Lindner[1], and Sascha Beutel[1]

Abstract

The introduction of smart virtual assistants (VAs) and corresponding smart devices brought a new degree of freedom to our everyday lives. Voice-controlled and Internet-connected devices allow intuitive device controlling and monitoring from all around the globe and define a new era of human–machine interaction. Although VAs are especially successful in home automation, they also show great potential as artificial intelligence-driven laboratory assistants. Possible applications include stepwise reading of standard operating procedures (SOPs) and recipes, recitation of chemical substance or reaction parameters to a control, and readout of laboratory devices and sensors. In this study, we present a retrofitting approach to make standard laboratory instruments part of the Internet of Things (IoT). We established a voice user interface (VUI) for controlling those devices and reading out specific device data. A benchmark of the established infrastructure showed a high mean accuracy (95% ± 3.62) of speech command recognition and reveals high potential for future applications of a VUI within the laboratory. Our approach shows the general applicability of commercially available VAs as laboratory assistants and might be of special interest to researchers with physical impairments or low vision. The developed solution enables a hands-free device control, which is a crucial advantage within the daily laboratory routine.

Keywords

virtual assistant, laboratory automation, voice user interface, Internet of Things, digital transformation

Introduction

Not long ago, the concept of a virtual assistant (VA) was something one would have surely imagined being set into the territory of science fiction, where artificial intelligence (AI)-driven entities support comic superheroes saving the day and help TV spaceship crews exploring places "no man has gone before." In recent years, natural language processing (NLP)—the foundation for voice-based VAs—has become more and more sophisticated and reliable.[1] Lately, many big information technology (IT) companies have started to work on VAs, which use speech recognition technology, AI, and speech synthesis in order to understand and answer questions and execute specific tasks.[2] Today smart personal assistants are commercially available and have found their way into our homes and pockets. When Apple released one of the first widely available VAs, Siri, to the public in late 2011, the masses were thrilled, although privacy and security concerns were raised immediately.[3] Soon other smart assistants were released: Google's Google Assistant, Microsoft's Cortana, Amazon's Alexa, and Samsung's Bixby, just to name the most popular.[4,5] Some VAs allow the extension of their capabilities by the installation of custom-developed applications, so-called "skills" or "actions." Besides use for entertainment and home automation purposes, there is also a demand to use the current VA technology in a professional environment. With speech being one of the most natural and intuitive means of interaction, machine control or data access using a voice interface offers advantages over conservative device interaction methods that are provided by human interface devices

[1]Institute of Technical Chemistry, Leibniz University Hannover, Hannover, Germany
[2]Institute of Brewing and Beverage Technology, Forschungszentrum Weihenstephan, Technische Universität München, Germany

Received Jan 23, 2018, and in revised form May 1, 2018. Accepted for publication June 20, 2018.

Supplemental material is available online with this article.

Corresponding Author:
Sascha Beutel, Institute of Technical Chemistry, Leibniz University Hannover, Callinstr. 5, 30167 Hannover, Germany.
Email: beutel@iftc.uni-hannover.de

(HIDs).[6] A hands-free machine control while pursuing other tasks, an intuitive interaction that does not require extensive training, and an easy device interaction for operators with visual or physical impairments are just a few benefits of a voice user interface (VUI).

So far, the use of VAs has been evaluated in different professional fields. Allen et al. used an Amazon Echo with a custom skill for a hands-free request of visual support on a tablet within a clinical study, while Miehle et al. presented a concept for VAs as a support in surgical operating rooms.[7,8] A smart cabinet with a VUI deploying an Amazon Echo in a custom service environment was developed by Ennis et al. as a support for the elderly in everyday situations.[9]

Also, a skill for assistance in chemistry labs was developed recently. "Helix" is an Amazon Echo-based skill that can look up chemical reaction information and chemical-associated data.[10] While the introduction of VAs into natural sciences just recently gained the interest of the scientific community, which is most probably due to the recent emergence of the technology and software, other innovative automation concepts and smart applications found their way into laboratories some time ago. Smart devices are already used to support scientists within their experiments. Current research topics include the use of smartphones, tablets, and smartglasses as inexpensive substitutions for some laboratory devices, for documentation tasks, and for data accession and analysis.[11–13] Novel automation efforts comprise the fully automated design, execution, and evaluation of experiments, which could lead to a tremendous efficiency enhancement of lab work in the near future.[14,15] The digital transformation of science has just started, but it seems to hold great potential to ease the work in all parts of scientific research.

The aim of this study was to establish a laboratory VUI for controlling laboratory instruments. This was achieved using commercially available hardware, cloud services, and open-source solutions. The setup is modular and allows the integration of older laboratory instruments. It showed high accuracy in the execution of speech commands. Using open communication protocols and data formats allows integration into available digital laboratory infrastructures.

Materials and Methods

Device IoT Retrofitting Approach

To integrate standard laboratory instruments into a modern Internet of Things (IoT) environment, the visual programming tool Node-RED 0.15.2[16] (IBM, Armonk, NY) deployed on a desktop PC (Intel i5 6500T CPU, 8 GB of RAM) running Windows 7 Professional 64-Bit OS (Microsoft Corporation, Redmond, WA) was used. Both laboratory devices utilize an RS232 interface for serial communication that was connected using an RS232-USB adapter. To realize an IoT capability, the proprietary port commands for device interaction were synchronized with the cloud-based IoT broker AWS IoT[17] (Amazon Web Services, Seattle, WA) that hosts a digital representation, a so-called device shadow, of the specific laboratory instrument. Port commands of the high-precision balance ED224S (Sartorius AG, Göttingen, Germany) were provided by the manufacturer's manual. Port commands and drivers for the spectrophotometer GENESYS 10S UV-Vis (Thermo Fisher Scientific Inc., Waltham, MA) were kindly provided by the manufacturer. The serial port/device shadow synchronization was implemented using JavaScript and standard function nodes in Node-RED. The bidirectional device shadow communication was realized using the MQTT (Message Queue Telemetry Transport) messaging protocol.[18] A secured MQTT connection between the device server and the specific device shadow was established using X.509 certificates over the designated secure MQTT TCP (Transmission Control Protocol) port 8883.[19] Node-RED standard nodes were used for MQTT-RS232/RS232-MQTT communication and string manipulation. Integration of a Wi-Fi socket (Edimax SP-2101W, Edimax Computer Company, Santa Clara, CA) for instrument power control was achieved with the custom node "node-red-contrib-smartplug 2.0.0." The device shadows were designed to perform all basic device interactions. The underlying JavaScript Object Notification (JSON) format is modular, which allows the addition of further device parameters at a later time (see **Fig. 2**).

Setup of a Custom Skill for Laboratory Instrument Interaction

An on-demand computing service[20] (Amazon Lambda, Amazon Web Services) was utilized to host a custom skill that interacts with the laboratory device shadows and reacts to processed speech command inquiries. Node.js 6.10 was used as a runtime environment and custom intents were implemented to realize specific device interactions. The Alexa Skills Kit (Amazon Web Services), which comprises the so-called "utterances"—custom phrases to invoke a specific device action, was enabled as a trigger for the hosted skill. An audio response model was implemented to give feedback to specific phrases.

Creation of a Voice Interaction Interface

The VUI was designed using the Alexa Skills Kit. The invocation names for the specific skills were defined as "photometer" and "balance," respectively. The corresponding endpoints were set to the specific Lambda application resource. The interaction model was designed redundantly to have a selection of different phrases triggering a specific

intent and resulting in an instrument action. The underlying speech recognition algorithms are part of Amazon's Alexa Voice Service[21] (AVS, Amazon Inc., Seattle, WA). An Amazon Echo first-generation device was used as a VA-enabled device. The primary language of the system was set to English.

Results and Discussion

Virtual Assistant Ecosystem

The voice model holds information about the skill's invocation name, for example, "photometer," which needs to be said to call a skill. The model also contains a defined phrase, for example, "to measure current value" or "to set wavelength to XXX nanometers." Specific phrases invoke an intent, which, regarding the abovementioned examples, causes a photometric measurement or sets the spectrophotometer to a desired wavelength (see **Fig. 1** for a dataflow scheme). This so-called interaction model uses an Internet-accessible skill hosting service as an endpoint and uses commercial NLP algorithms (Amazon) to process speech inquiries and generate speech output. The skill accepts requests based on the interaction model and performs actions upon these, for example, providing an audio feedback and/or interacting with the device shadow that is a digital representation of the current device state. The device shadow basically acts as an MQTT broker that holds a predefined key–value JSON of the device's conditions and states (see **Fig. 2** for the JSON layout of the used device shadows).

This shadow can be accessed and modified by the skill. This can, for example, set the value of the subkey "active" to "true" within the "measuring" key after the specific intent is called. The shadow can also be accessed by the local service, which can update the device shadow values for a defined key, for example, writing a measured absorbance value to a defined turret position key after an absorption is measured or setting "measuring" to "false" after a successful measurement. Specific limits were defined for specific values; for example, the spectrophotometer wavelength can only be set between 190 and 1100 nm. If the operator wishes to set up a wavelength out of this range, the VA-enabled device gives out a failure message and specifies the valid values.

A complete speech command consists of a wake-up word, which is "computer" in our setup; the name of the skill, which is either "the photometer" or "the balance"; and a phrase that is mapped to an action. A total of 18 phrases were defined for seven different interactions with the photometer, whereas 9 phrases for four different balance interactions were implemented (see **Table 1** for example phrases).

Due to its modular fashion, the developed environment allows for easy integration of additional speech commands by only adding compatible code on the cloud side of the system. The integration of a similar device from another vendor is possible. Only the specific proprietary interaction commands and the communication standard (depending on the installed data exchange interface) within the local service need to be changed. The integration of other laboratory device types is possible. A reasonable device shadow design needs to be worked out for every new device type, and a new voice model and custom skill need to be implemented. The use of an open communication protocol (MQTT) and an open and compact data format (JSON) allows the integration of accruing data into laboratory information management systems (LIMSs) or enterprise resource planning (ERP) systems. Transport Layer Security (TLS) encryption provides advanced data security for transmissions from the cloud to the local service environment and vice versa. Since data security is always a major topic in science, engineering, and research, the established infrastructure might be seen as critical. The speech recognition is done by an external service provider, and all arising data are stored on an external service provider's servers, which can always be targeted by attacks. From the privacy point of view, the smart-enabled device records voice after a specific wake-up word is said. It can potentially be cracked, which can grant unauthorized access to the connected device's capabilities.[4,22] The use of the system by unauthorized personnel could potentially be avoided by using voiceprint identification, which can limit the VA features to a specific group of users.[23,24] A function like this might be implementable in the not too distant future.[25]

Voice User Interface Accuracy

To determine the accuracy of the VUI and the speech command recognition capability of the underlying cloud algorithm, each of the particular phrases for the device control of the spectrophotometer or the balance was said 30 times in a row by a male scientist, who was already familiar with the developed system and the defined phrases. It was then checked whether the given speech commands resulted in the expected device actions. The evaluation was done in a laboratory with active exhaust ventilation that caused ambient noise in the range of 50–55 dB, which is equivalent to the sound level of a normal conversation. The speech commands were given 1.5 m from the VA-enabled device. Possible evaluation outcomes were that the system did not map the phrases to the correct intent, maybe due to an ambiguity error; that it could not connect the said phrases to any intent, due to not understanding the phrase; or that it mapped the specific phrase to the correct intent. The first two scenarios were both graded as "not correctly recognized," while the last was marked as "correctly recognized." The command recognition accuracy analysis resulted in an overall mean recognition accuracy of 95% ± 3.62 (see **Table 1**

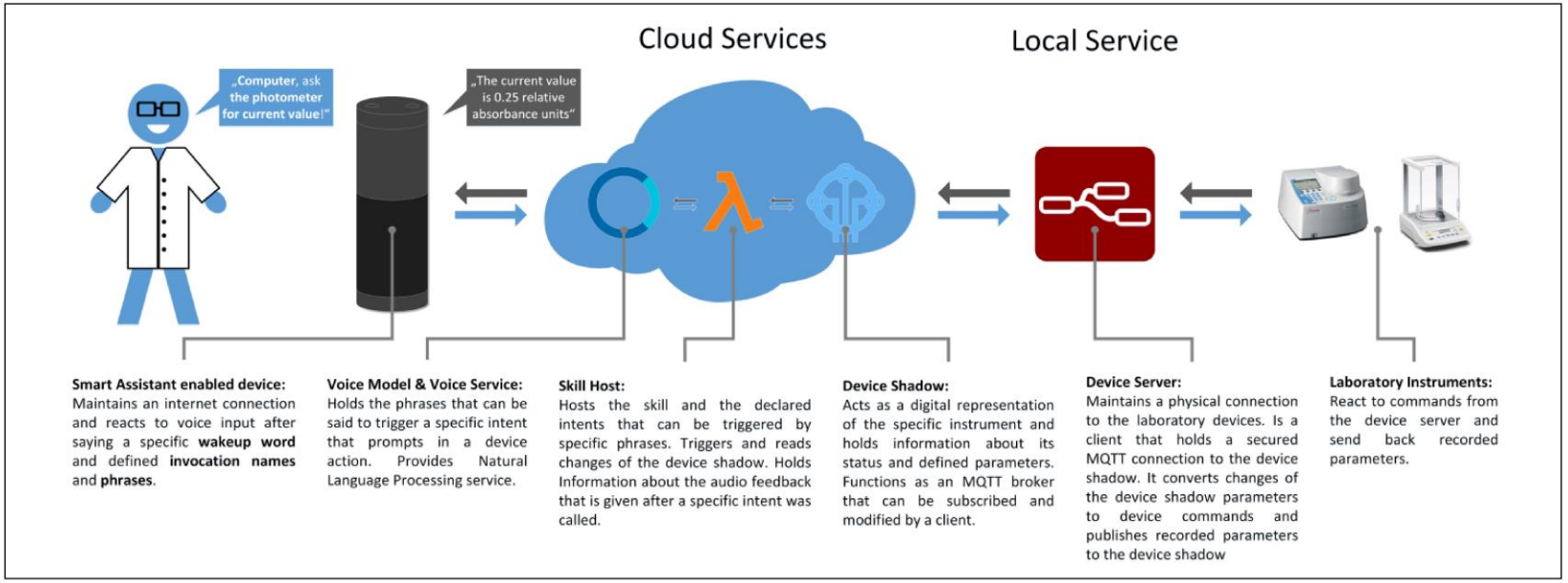


Figure 1. Established VA ecosystem. The figure shows the dataflow of the system. First, a defined speech command is given by the operator that gets recognized and recorded by a VA-enabled device and is then processed by a cloud voice service. Afterwards, a mapped intent within a cloud-hosted skill is called that modifies the device shadow of an integrated laboratory instrument. Changes of a device shadow get recognized by a local service and result in a device action. Taken measurements or parameter changes get recorded, and the device shadow gets changed by a local service that functions as a client of the device shadow service. This changed information is called by the skill, which provides an audio feedback that is processed by the voice service and given out by the VA-enabled device.

A


```
{
   "power": true,
   "wavelength": "340",
   "measuring": {
      "active": false,
      "values": {
         "0": 0.0001,
         "1": 0.1702,
         "2": 0.0004,
         "3": 0.0947,
         "4": 0.0001,
         "5": 0.0001
      }
   },
   "beeping": false,
   "blanking": false,
   "turret": 3
}
```


B


```
{
   "measuring": {
      "active": false,
      "value": 43.8805
   },
   "taring": false,
   "blocked": false
}
```


Figure 2. Specified device shadows of the integrated laboratory devices. (**A**) JSON of the photometer shadow. Key–value pairs that handle the power supply of the device, the wavelength, the measure function, and the last measured absorption values of the photometer of the corresponding turret positions. Also, the acoustic signal function, blanking function, and current sample position are implemented. (**B**) JSON of the high-performance balance. Key–value pairs of the JSON depict the measuring function, the last measured value, the taring function, and the blocking and unblocking of the physical device buttons.

for details and supplemental material for a detailed listing of all implemented phrases and video material of the specific device interactions). This accuracy seems to be adequate, taking into account that the operator has a German accent, which might influence the voice recognition success rate.[26] It has to be mentioned that the analyzed setup was already established for 4 months at the moment of the accuracy determination, and that it was trained by male as well as female users, with no perceptible differences concerning the voice recognition accuracy between the sexes. The experiences from this training process led to the suggestion that the recognition accuracy might be similar with different users, at least if they have the same accent. The used NLP service gets trained perpetually by using data from speech interactions, which means that the current shown data are just a snapshot, as the service gets more efficient every day. The high efficiency of the used cloud service demonstrates the capability of commercially available NLP algorithms and highlights that an application in professional environments is already imaginable.

The digital transformation of industry and everyday life is eminent and will pervade most facets of society. This new era of smart devices and services offers tremendous possibilities to support engineers and scientists. As demonstrated in this report, utilizing virtual voice-based assistants in the laboratory might be a powerful tool to improve the efficiency of laboratory processes.

We were able to integrate present laboratory devices into an IoT environment in an affordable way. By using a combination of commercially available and open-source solutions, a virtual laboratory assistant was implemented. It acted as a voice-based user interface to laboratory instruments. Although all data transfer within this system is encrypted, potential data security and privacy threats might arise due to the external processing and generation of speech data. NLP is a hardware-demanding task but can potentially be established as an in-house-solution to bypass potential security concerns. This would require substantial know-how and maintenance expenses, though, whereas the current generation of skills requires Internet-accessible endpoints. Also, a stable and fast Internet connection is needed to access the used cloud services, which might not always be given.

The use of voice commands and the VA voice output adds to the general cacophony in the lab and might be distracting to other laboratory staff. The pairing with Bluetooth headphones can already be accomplished with some Echo hardware versions (e.g., the Echo Dot), which might minimize noise annoyance caused by VA voice output. Amazon recently announced the “Alexa Mobile Accessory Kit” to

Table 1. Examples of Implemented Speech Commands for Device Interaction and the Mean Recognition Accuracy of All Implemented Phrases That Cause the Same Action (see supplemental material for other implemented phrases).

Device	Example Phrase	Action	Mean Recognition Accuracy (%)
Photometer	"To beep"	Photometer gives acoustic signal	91.7
Photometer	"To get current value"	Photometer measures and VA gives out current value	92.8
Photometer	"To go to sample position X"	Photometer turret moves to desired turret position	91.7
Photometer	"To turn on"	Photometer turns on	98.3
Photometer	"To turn off"	Photometer turns off	95
Photometer	"To blank"	Photometer blanks	98.3
Photometer	"To set wavelength to XXX nanometers"	Photometer grate is set to desired wavelength	93.3
Balance	"To return weight"	Balance measures and VA gives out current weight	97.8
Balance	"To zero"	Balance tares	95
Balance	"To block keys"	Balance buttons get blocked	98.3
Balance	"To unblock keys"	Balance keys get unblocked	96.7

bring Alexa to on-the-go devices like headsets, headphones, and smartwatches.[27] This offers the chance to add mobility to the VA and decrease the noise level within the laboratory.

VA technology and software is not limited to the presented VA-enabled device class. AI-enabled smartglasses were recently announced that might form a fruitful symbiosis between safety goggles, which are mandatory in the laboratory; a virtual laboratory assistant; and augmented reality elements.[28] This combination could be a great addition to the laboratory workflow (e.g., to interactively display device parameters, standard operating procedures [SOPs], and safety-relevant notifications in the experimenter's field of view). Since cloud NLP is not as limited as the integrated voice recognition technology that most current smartglasses use for voice handling, even complex voice commands could be processed. Yet current hardware and battery limitations of smartglasses need to be overcome to integrate the technology into everyday working practice, but they offer another way to bring AI to the lab.[29]

To conclude, it can be said that VUIs hold great potential for applications within the laboratory—not only as a means for instrument control, but also as a system to retrieve and save experimental data and protocols. The proof-of-concept setup shown in this study is surely only one of the first steps toward an AI-assisted laboratory. It indicates that present digitalization and automation concepts have the potential to improve the workflows and efficiency in the laboratory.

Current efforts of laboratory instrument manufacturers are focused on the development of standardized formats and protocols for laboratory device communication. This might pave the way to a plug-and-play IoT capability of future laboratory instruments.[30] The shown concept seems to be a useful addition to the laboratory. In order to find a place for a VA in the lab environment of the future, however, security solutions that reflect common privacy and confidentiality rules need to be developed.

Our ongoing research is focused on implementing further instrument types and the integration of the virtual laboratory assistant into a functional laboratory information and management system. Also, the implementation of SOPs supported by a VA is planned.

Declaration of Conflicting Interests

The authors declared no potential conflicts of interest with respect to the research, authorship, and/or publication of this article.

Funding

The authors disclosed receipt of the following financial support for the research, authorship, and/or publication of this article: This work was financially supported by the Bavarian Ministry of Economic Affairs and Media, Energy and Technology within the Information and Communications Technology program (grant number IUK-1504-0006//IUK470/001).

ORCID iD

Jonas Austerjost https://orcid.org/0000-0002-2080-1556

References

1. Cambria, E.; White, B. Jumping NLP Curves: A Review of Natural Language Processing Research. *IEEE Comput. Intell. Mag*. **2014**, *9*, 48–57.
2. Dale, R. The Commercial NLP Landscape in 2017. *Nat. Lang. Eng*. **2017**, *23*, 641–647.
3. McMillan, R. IBM Worries iPhone's Siri Has Loose Lips. CNN, May 24, 2012. http://edition.cnn.com/2012/05/23/tech/mobile/ibm-siri-ban/index.html (accessed Nov 18, 2017).
4. Dale, R. Industry Watch: The Pros and Cons of Listening Devices. *Nat. Lang. Eng*. **2017**, *23*, 969–973.
5. Gopinath, S. C. B.; Tang, T. H.; Chen, Y.; et al. Bacterial Detection: From Microscope to Smartphone. *Biosens. Bioelectron*. **2014**, *60*, 332–342.

6. Cohen, M. H.; Giangola, J. P.; Balogh, J. Voice User Interface Design. *Computer* **2004**, *49*, 368.
7. Allen, A. A.; Shane, H. C.; Schlosser, R. W. The Echo™ as a Speaker-Independent Speech Recognition Device to Support Children with Autism: An Exploratory Study. *Adv. Neurodev. Disord*. **2017**, 1–6.
8. Miehle, J.; Ostler, D.; Gerstenlauer, N.; et al. The Next Step: Intelligent Digital Assistance for Clinical Operating Rooms. *Innov. Surg. Sci*. **2017**, *2*, 159–161.
9. Ennis, A.; Rafferty, J.; Synnott, J.; et al. A Smart Cabinet and Voice Assistant to Support Independence in Older Adults. In *Lecture Notes in Computer Science*; Springer: Cham, 2017; Vol. 10586 LNCS, pp 466–472.
10. Halford, B. Meet Your New Lab Assistant. *Chemical and Engineering News*, 2017, pp 26–27.
11. Young, H. A. Scientific Apps Are Here (and More Will Be Coming). *Cytokine* **2012**, *59*, 1–2.
12. Gan, S. K.-E.; Poon, J.-K. The World of Biomedical Apps: Their Uses, Limitations, and Potential. *Sci. Phone Apps Mob. Devices* **2016**, *2*, 6.
13. Austerjost, J.; Marquard, D.; Raddatz, L.; et al. A Smart Device Application for the Automated Determination of *E. coli* Colonies on Agar Plates. *Eng. Life Sci*. **2017**, *17*, 959–966.
14. Wu, T.; Zhou, Y. An Intelligent Automation Platform for Rapid Bioprocess Design. *J. Lab. Autom*. **2014**, *19*, 381–393.
15. Glauche, F.; Pilarek, M.; Bournazou, M. N. C.; et al. Design of Experiments-Based High-Throughput Strategy for Development and Optimization of Efficient Cell Disruption Protocols. *Eng. Life Sci*. **2017**, *17*, 1166–1172.
16. IBM Foundation. Node-RED: Documentation. https://nodered.org/docs/ (accessed Jan 22, 2018).
17. Amazon Web Services. AWS IoT Core—Dokumentation. https://aws.amazon.com/de/documentation/iot/ (accessed Jan 22, 2018).
18. Hunkeler, U.; Truong, H. L.; Stanford-Clark, A. MQTT-S—A Publish/Subscribe Protocol for Wireless Sensor Networks. In *2008 3rd International Conference on Communication Systems Software and Middleware and Workshops (COMSWARE '08)*; IEEE: Piscataway, NJ, 2008; pp 791–798.
19. Reinfurt, L.; Breitenbücher, U.; Falkenthal, M.; et al. Internet of Things Patterns. In *Proceedings of the 21st European Conference on Pattern Languages of Programs—EuroPlop '16*; ACM Press: New York, 2016; pp 1–21.
20. AWS Lambda—Serverless Computer—Amazon Web Services. https://aws.amazon.com/lambda/?nc1=h_ls (accessed May 1, 2018).
21. Amazon Developer. Alexa Voice Service Overview. https://developer.amazon.com/de/alexa-voice-service (accessed Jan 22, 2018).
22. Chung, H. "Alexa, Can I Trust You?" *Computer (Long Beach Calif.)* **2017**, *50*, 100–104.
23. Doddington, G. R. Speaker Recognition—Identifying People by Their Voices. *Proc. IEEE* **1985**, *73*, 1651–1664.
24. Poddar, A.; Sahidullah, M.; Saha, G. Speaker Verification with Short Utterances: A Review of Challenges, Trends and Opportunities. *IET Biometrics* **2018**, *7*, 91–101.
25. Eadicicco, L. Exclusive: Amazon Developing Advanced Voice-Recognition for Alexa. *Time*, Feb 27, 2017. http://time.com/4683981/amazon-echo-voice-id-feature-2017/ (accessed Mar 15, 2018).
26. Huang, C.; Chen, T.; Chang, E. Accent Issues in Large Vocabulary Continuous Speech Recognition. *Int. J. Speech Technol*. **2004**, *7*, 141–153.
27. Luthra, G. Amazon Alexa Mobile Accessories: A New Alexa-Enabled Product Category with Dev Tools Coming Soon. *Alexa Blogs*, Jan 5, 2018. https://developer.amazon.com/de/blogs/alexa/post/564685cf-0b1b-4fe4-824e-2ce1e88e3e78/amazon-alexa-mobile-accessories-a-new-alexa-enabled-product-category-with-dev-tools-coming-soon (accessed May 1, 2018).
28. Stein, S. Alexa-Powered Smartglasses? Yes, from Vuzix. *CNET, Jan 16*, 2018. https://www.cnet.com/news/vuzix-debuts-amazon-alexa-blade-smartglasses-ces-ar/ (accessed May 1, 2018).
29. Martinez-Millana, A.; Bayo-Monton, J. L.; Lizondo, A.; et al. Evaluation of Google Glass Technical Limitations on Their Integration in Medical Systems. *Sensors (Switzerland)* **2016**, *16*, 2142.
30. Schmid, I.; Aschoff, J. A Scalable Software Framework for Data Integration in Bioprocess Development. *Eng. Life Sci*. **2017**, *17*, 1159–1165.

3.4 BETRACHTUNG DER MÖGLICHKEITEN DES EINSATZES VON SMARTGLASSES IM LABOR

Als Interaktionsmedium des digitalisierten Labors mit dem Verwender bieten sich besonders SmartGlasses an. Dabei handelt es sich um so genannte „*head-mounted-displays*“, also am Kopf befestigte Bildschirme. Diese werden für den Einsatz im Labor mit Hilfe von speziellen Halterungen an der ohnehin obligatorischen Laborschutzbrille befestigt. Damit befinden sich diese Displays kontinuierlich im Sichtfeld des Verwenders und können eingesetzt werden, um kurze Informationsbausteine kontextsensitiv einzublenden. Viele SmartGlass-Modelle verfügen außerdem über eine Spachsteuerung, die das Bedienen der Geräte freihändig möglich macht.

Der folgende Artikel evaluiert die Einsatzmöglichkeiten von SmartGlasses im Laborumfeld und stellt neben den technischen Möglichkeiten und Einsatzbeispielen auch die rechtlichen und gesundheitlichen Rahmenbedingungen dar. Beispielsweise werden Einsatzszenarien gezeigt, in denen die SmartGlasses unterstützende Informationen für den Anwender bereitstellen (siehe z. B. Abbildung 24) oder auch als Interaktionsmedium mit einem digitalen Backendsystem fungieren.

So wird ein Beispiel-Arbeitsablauf gezeigt, in dem SmartGlasses zur Ergebnisdateneingabe bei einer Pflanzenbonitur verwendet werden. Die verschiedenen verfügbaren SmartGlass-Modelle zeichnen sich meist durch eine einfache Programmierbarkeit aus, da sie die gleiche Hardware aufweisen wie verbreitete Smartphones oder Tablet-Computer. Dadurch können mit Hilfe der für diesen Einsatzzweck bereits gut etablierten Werkzeuge, Applikationen für die Anwendung auf SmartGlasses entwickelt werden.

Auf diese Art werden SmartGlasses ebenfalls im hier vorgestellten Labordigitalisierungskonzept angebunden. Eine Applikation empfängt Informationen vom Laborserver und stellt diese grafisch aufbereitet dem Benutzer dar. Nutzereingaben werden auf umgekehrtem Weg an das System übertragen. Da die verwendeten SmartGlasses, Tablets und Oberflächen-Projektoren alle Google Android Betriebssysteme besitzen, sind die entwickelten Applikationen gut portabel und die Integration weiterer Endgeräte stellt keine Herausforderung dar.

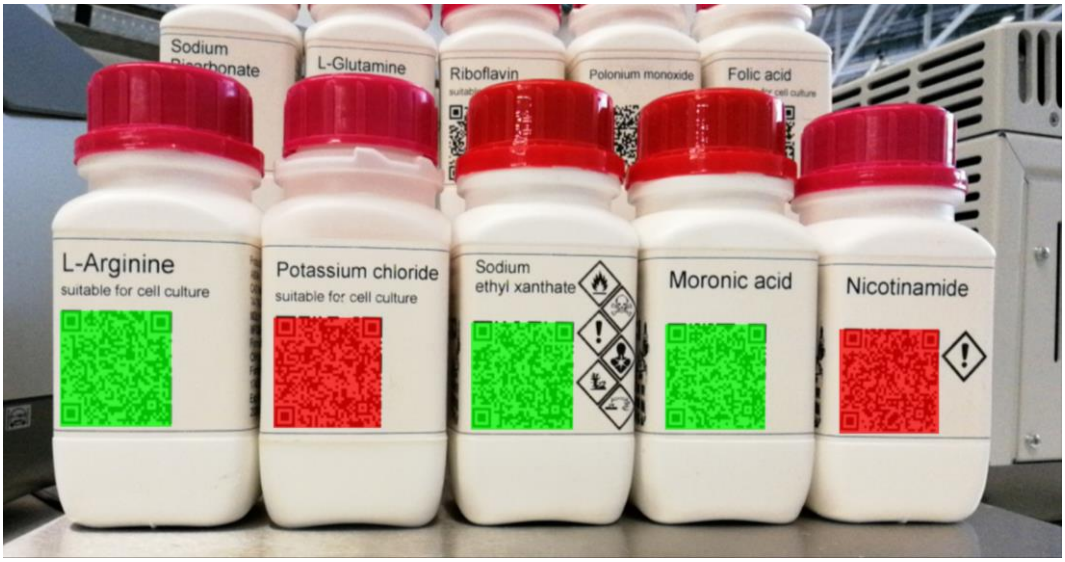


ABBILDUNG 24: UNTERSTÜTZUNG DES LABORANTEN DURCH SMARTGLASSES. DIE GEZEIGTEN CHEMIKALIENGEBINDE SIND DURCH QR-CODES GEKENNZEICHNET UND ENTSPRECHENDE INFORMATIONEN ZU DEN INHALTEN IN EINER INVENTARDATENBANK HINTERLEGT. IM GEZEIGTEN FALL WERDEN ABGELAUFENE CHEMIKALIEN OPTISCH DURCH ROTFÄRBUNG ANGEZEIGT.

CHEMIE IN UNSERER ZEIT
DOI: 10.1002/ciuz.2020

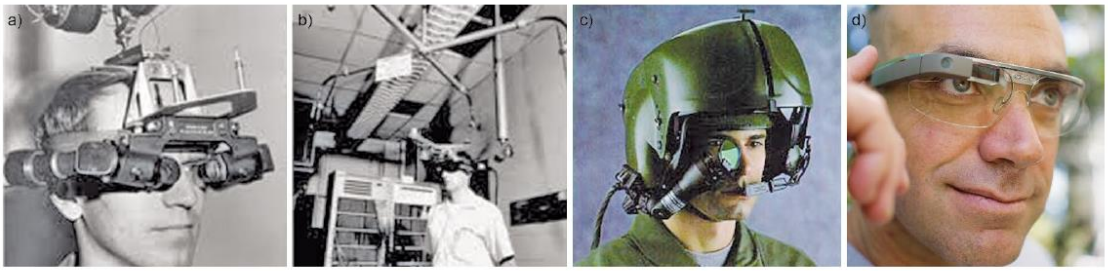


Abb. 1 *(a, b) Das weltweit erste Head Mounted Displays (HMD) mit dem Namen „Schwert des Damokles" (1968). (c) Eines der ersten produktiv genutzten HMDs, das „Integrated Helmet and Display Sighting System" (IHADSS) für Apache Kampfhubschrauber (1985). (d) Google Glass® (2013)* [5–7].

Werkzeug oder Spielzeug?

Smartglasses im Labor

DANIEL MARQUARD | MARC PORR | FERDINAND LANGE | JONAS AUSTERJOST | SASCHA BEUTEL

Digitalisierung findet mehr und mehr in allen Arbeitsbereichen Nutzen und Verwendung. Dabei wird allerdings häufig vergessen, dass der Mensch auch in angemessener Weise mit der digitalen Infrastruktur interagieren können muss. Hierfür eignen sich prinzipiell viele Medien, zum Beispiel das Smartphone oder Tablets. In Arbeitsbereichen, in denen etwa mit Chemikalien gearbeitet wird, wie im Labor, stoßen diese allerdings schnell an ihre Grenzen, denn mit Handschuhen sind sie nur schwer zu bedienen. Eine Lösung hierfür könnten sogenannte Wearables sein, und hier vor allem die Smartglasses, da im Labor aus Gründen der Arbeitssicherheit sowieso Schutzbrillen getragen werden müssen und diese einfach um eine Datenbrillenfunktion erweitert werden könnten. Die Chancen, Möglichkeiten und Risiken des Einsatzes von Smartglasses im Laborbereich sind aktuell Gegenstand der Forschung.

Die Idee, grafische Informationen direkt in das Sichtfeld eines Menschen einblenden zu können und so eine Brücke zwischen realer und virtueller Welt zu schaffen, hat einige Menschen schon lange fasziniert. So entstand das erste Head Mounted Display (HMDs) schon Ende der 1960er Jahre, also vor mehr als 50 Jahren, durch Pioniere wie Ivan Sutherland. Das Gerät sollte am Kopf getragen werden und grafische Inhalte in das Sichtfeld des Benutzers einblenden [1]. Das von Sutherland und seinen Kollegen geschaffene Gerät wurde allerdings „Schwert des Damokles" genannt, da es aufgrund seines massiven Gewichts und seines sperrigen Kopfverfolgungsmechanismus an der Decke befestigt werden musste und bedrohlich über dem Anwender zu schweben schien (Abbildung 1a, b).

Über die letzten Jahrzehnte konnten aufgrund der Miniaturisierung von Elektronik und Optik immer kleinere und leistungsfähigere HMDs entwickelt werden. In den 1980er Jahren erreichten HMDs eine Größe, so dass sie an Helmen befestigt werden konnten und die dazugehörige Technik prinzpell in einen Rucksack passte [2]. So wurden ab 1985 die Piloten des Apache Kampfhubschraubers mit dem „Integrated Helmet and Display Sighting System" ausgestattet (Abbildung 1c), welches ihnen das Livebild einer Infrarotkamera direkt ins Sichtfeld einblendete [3]. Dadurch wurde den Piloten Nachtsicht ermöglicht und sie konnten sehen, was sich direkt unter ihrem Hubschrauber befand. Heute unterscheiden sich die Ausmaße von HMDs kaum noch von denen normaler Brillen und bilden damit die Grundlage für moderne Smartglasses [4].

Als im Jahr 2013 das Smartglasses-Modell Google Glass® eingeführt wurde, war die Smartglasses-Technologie nun erstmals auch einem breiteren Publikum im Verbraucherbereich zugänglich (Abbildung 1d). Andere Hersteller folgten Google und brachten Modelle mit ähnlichen technischen Eigenschaften heraus. Obwohl die meisten der aktuellen Smartglasses-Modelle im Verbraucherbereich nicht

Online-Ausgabe unter: wileyonlinelibrary.com

erfolgreich sind, wurden viele Anwendungen in professionellen Bereichen entwickelt. Diese umfassen Dokumentations-, Visualisierungs- und Schulungszwecke im Gesundheitswesen sowie Unterstützung von Arbeitsabläufen und Qualitätssicherung in Logistik und Fertigungsabläufen [8, 9].

Ein weiteres spannendes Einsatzgebiet für Smartglasses sind chemische und biologische Labore. Seit Jahren gib es Bestrebungen hin zum papierlosen, voll digitalisierten Labor, in dem alle Geräte miteinander vernetzt sind und alle Abläufe digital überwacht und dokumentiert werden [10, 11]. Aber ein voll digitalisiertes Labor ist kein voll automatisiertes Labor. Für viele Experimente und insbesondere für komplexe Experimente werden weiterhin Menschen im Labor benötigt. In der Konsequenz müssen die Menschen mit dem digitalisierten Labor interagieren können.

Dafür werden Interaktionsmedien benötigt, die die Schnittstelle zwischen Mensch und digitalem Labor bilden. In den meisten Fällen werden dafür aktuell Tablets aus dem Verbraucherbereich verwendet. Gründe dafür sind verhältnismäßig geringe Kosten und die hohe Akzeptanz, da die meisten Menschen Tablets auch privat verwenden. Allerdings haben Tablets das Problem, dass sie mit Händen im Labor bedient werden, was leicht zu chemischen und biologischen Kontaminationen führen kann. Diese Kontaminationen sind nur schwer zu entfernen, da die meisten Reinigungsmittel das Display und/oder die Elektronik angreifen können. Außerdem ist es für viele Abläufe im Labor von Vorteil, beide Hände zur Verfügung zu haben.

Deshalb werden aktuell sinnvolle Alternativen für den Einsatz im Labor gesucht und erprobt. Porr *et al.* schlagen die Verwendung von Touchbeamern im Labor vor, diese funktionieren wie ein Tablet, allerdings wird das Bild auf die Tischoberfläche projiziert und mit einem Infrarotraster überlagert, welcher Berührungen registriert. Dadurch wird die Tischoberfläche zum Bedienfeld und das Gerät selber ist gegen Kontaminationen geschützt [10]. Austerjost *et al.* wiederum haben unter Verwendung eines Amazon Echo Smartspeakers eine rein sprachbasierte Geräteinteraktion entwickelt [12]. Jedoch wird in diesem Konzept komplett auf eine Visualisierung verzichtet, was es dem Menschen im Labor schwierig macht, komplexe Abläufe zu erfassen. Außerdem sind weder beim Touchbeamer noch beim Smartspeaker eine Kamera integriert, die für die Dokumentation im Labor immer wichtiger wird. So ist ersichtlich, dass es weiteren Bedarf für geeignete Interaktionsmedien in Laboren gibt. Und hier kommen die Smartglasses ins Spiel.

In den meisten Laboren müssen aus Gründen der Arbeitssicherheit Schutzbrillen verpflichtend getragen werden. Warum also diesen Schutzbrillen nicht einen zusätzlichen Mehrwert geben? Smartglasses verfügen über eingebaute Sensoren wie Mikrophone und Kameras (Abbildung 2). Diese können für die Dokumentation von Experimenten verwendet werden, aber auch als Messsystem. So können aufgenommene Bilder direkt von Bildverarbeitungsalgorithmen ausgewertet werden. Darüber hinaus verfügen Smartglasses über WLAN- und Bluetooth-Schnittstellen, die zum einem die Anbindung von externen Sensoren und Geräten an die Smartglasses erlauben und zum anderen die Einbettung und den Transfer von Daten in eine übergeordnete IT-Infrastruktur ermöglichen. Über das Display der Smartglasses können Informationen, Arbeitsanweisungen oder Messergebnisse direkt ins Sichtfeld eingeblendet werden. Diese Informationen können mit Live-Bildern aus der Smartglasses-Kamera überlagert werden, um die Qualität der Informationen noch weiter zu erhöhen. Diese Arten der Informationsdarstellungen werden als Assisted- bzw. Augmented Reality bezeichnet (Infokasten 1). Die meisten Smartglasses verfügen zudem über eine Sprachsteuerung, über die der Mensch im Labor mit dem digitalen System interagieren kann und dennoch beide Hände für seine Arbeit im Labor frei hat.

Im Folgenden werden einige Anwendungen für Smartglasses vorgestellt, die entweder bereits veröffentlicht sind oder aus aktuellen Forschungsarbeiten stammen. Dabei wird auf Anwendungen monookularer Systeme, wie in Abbildung 2 dargestellt, fokussiert, da diese Systeme aufgrund ihres geringen Gewichtes und Tragekomforts am besten für die Arbeit im Labor geeignet erscheinen. Andere Systeme wie die Augmented Reality-Brillen (Infokasten 1 mittleres Bild) sind aktuell noch zu schwer und schränken das Sichtfeld zu stark ein, um sie sicher im Labor zu verwenden.

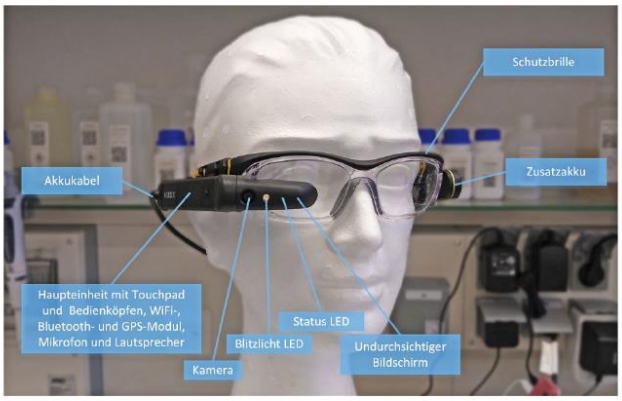


Abb. 2 ***Smartglasses mit Schutzbrillenfunktion (Vuzix M300)***

Digitale Arbeitsabläufe

In allen zertifizierten Laboratorien, zum Beispiel in der Umwelt- oder Lebensmittelanalytik, werden heute Standard-Arbeitsabläufe, Standard Operation Procedures (SOPs) verwendet. Hierbei handelt es sich um hochstrukturierte, kleinschrittige Anleitungen für einen Arbeitsablauf oder ein Experiment, aktuell zumeist noch papierbasiert. Durch die Entwicklung von digitalen Infrastrukturen im Labor können diese papierbasierten SOPs durch digitale SOPs

1: VERSCHIEDENE EBENEN DER VIRTUALISIERUNG

Abb. Info 1-1 ***Beispiele für Brillen-Systeme und Darstellungen für die Virtualisierung von Virtual Reality (VR), Augmented Reality (AR) und Assisted Reality (AsR). Die Darstellung entspricht dem, was die Person, die das Brillen-System verwendet, sehen würde. VR: Oculus Rift von Sony. Virtuelle 3D-Darstellung eines futuristischen Laborkonzeptes*** *[25]*. ***AR: Hololens von Microsoft*** *[26]*. ***Positionsgetreue Einblendung von Gefäßverläufen auf eine Übungspuppe für die Vereinfachung von medizinischen Anwendungen*** *(modifiziert nach [27])*. ***AsR: M300 von Vuzix. Einblendung von Komponenten bei der Herstellung eines Mediums.***

Der Übergang zwischen realer und virtueller Welt wird immer fließender, besonders bei der Verwendung von Smartglasses und anderen Head-Mounted Displays. Es gibt unterschiedliche Ebenen, die den Grad der Virtualisierung beschreiben. Wir schlagen eine Unterteilung in die drei Stufen Virtual Reality (VR), Augmented Reality (AR) und Assisted Reality (AsR) vor (Abb. Info 1-1), da diese auf Anwendungsebene eine gute Abgrenzung zwischen den Bereichen ermöglichen.

Bei der ***Virtual Reality*** *setzt der Mensch eine VR-Brille auf, die ihn visuell komplett von der realen Welt abschirmt, und er bekommt nur noch virtuelle Inhalte angezeigt. VR-Anwendungen sind deshalb für den direkten Einsatz im Labor nicht geeignet. Dafür eignen sie sich sehr gut für Trainings-, Planungs- und Visualisierungszwecke, da in der virtuellen Welt prinzipiell beliebig komplexe Vorgänge abgebildet werden können, ohne dass räumliche oder physikalische Limitierungen berücksichtigt werden müssten. So können Laboreinrichtungen virtuell geplant und auf Funktionalität getestet werden.*

Bei der ***Augmented Reality*** *wird auch von erweiterter Realität gesprochen. Hierbei wird die reale Umgebung durch das Tragen einer AR-Brille um eine digitale Schicht erweitert. Bei AR-Anwendungen wird die reale Welt mit virtuellen 3D-Inhalten positionsgetreu überlagert. Im optimalen Fall entsteht der Eindruck einer Verschmelzung zwischen realen und virtuellen Elementen. AR-Anwendungen haben ein großes Potential für die Laborarbeit, da sie den Menschen sehr gut visuell anleiten können und auch aufgrund der verbauten 3D-Kameras ein sehr gutes Feedback über die Ausführung geben können. Ein Beispiel wäre die korrekte Position eines Gefäßes oder einer Probe in der Zentrifuge. Der Einsatz von AR scheitert aktuell noch an zwei Problemen: Zum einem sind AR-Brillen relativ schwer und schränken das Sichtfeld ein, was das Tragen auf Dauer sehr anstrengend macht. Zum anderen sind AR-Inhalte aufwändig zu produzieren, da neben 3D-Modellen noch zusätzlich Modelle zur Interaktion mit der realen Umgebung benötigt werden.*

Unter ***Assisted Reality*** *wird das Einblenden von Informationen ins Sichtfeld des Menschen verstanden. Die meisten AsR-Brillen sind monookulare Systeme, also stellen Informationen nur auf einem Auge zur Verfügung. Mögliche Anwendungen sind das Anzeigen einer Medienrezeptur oder von Sicherheitshinweisen zu bestimmten Chemikalien. AsR-Anwendungen sind die aktuell am meisten erprobten Anwendungen für die Laborarbeit. AsR-Brillen sind im Vergleich zu AR-Brillen leichter und schränken das Sichtfeld kaum ein. Außerdem sind AsR-Anwendungen verhältnismäßig leicht zu entwickeln.*

Diese drei Ebenen sind nur eine grobe Einstufung zur Orientierung, es gibt besonders zwischen AR und AsR viele Überlappungen. So kann beispielsweise bei AR-Anwendung ein einfaches Textfeld eingeblendet werden. Andersherum kann bei einer AsR-Anwendung ein Livebild der Kamera angezeigt und auf diesem bestimmte Elemente wie QR-Codes farblich hervorgehoben werden (Abb. Info 1-2).

Abb. Info 1-2 ***Screenshot ChemFinder-App. In der ChemFinder-App wird das Livebild der Kamera dargestellt und gleichzeitig die Bilder nach QR-Codes durchsucht. Gefundene QR-Codes werden ausgelesen und zum Beispiel auf ein Ablaufdatum überprüft. QR-Codes von abgelaufenen Chemikalien werden mit einem roten Rechteck im Livebild überlagert, von noch nicht abgelaufenen mit einem grünen.***

ersetzt werden [10]. Ein prägnantes Beispiel liefern die von Austerjost *et al.* entwickelten digitalen SOPs zu MEBAK-Methoden; dabei handelt es sich um Standardanalysenmethoden der Mitteleuropäischen Brauereitechnischen Analysenkommission (MEBAK) zur Bewertung von Biercharakteristika wie der Bierfarbe, des Stickstoffgehaltes oder der Stammwürze.

Diese digitalen SOPs ermöglichen nun selbst Laien die Durchführung dieser Untersuchungen [11]. Denn die digitalen SOPs haben zum Ziel, den Menschen beim Arbeitsablauf möglichst umfassend zu unterstützen und so vor allem bei der Vermeidung von Fehlern zu helfen. Hierfür wird dieser schrittweise durch den Arbeitsablauf geführt und bekommt die relevanten Informationen strukturiert und immer aktuell für den jeweiligen Arbeitsschritt dargestellt. Optimalerweise werden die vernetzten Laborgeräte automatisch mit den Einstellungen und Messparametern programmiert und die digitale Infrastruktur übernimmt gleichzeitig auch die Dokumentation der Abläufe, so dass sich der Mensch voll auf die eigentliche Labortätigkeit konzentrieren kann. Dies reduziert die Fehlerwahrscheinlichkeit, macht aber die Erstellung von digitalen SOPs auch aufwändig im Vergleich zu papierbasierten SOPs (Infokasten 2).

Smartglasses sind für die Durchführung von digitalen SOPs ein mächtiges Werkzeug. Die aktuellen Informationen können auf dem Bildschirm der Smartglasses angezeigt werden. Dadurch hat der Mensch die Informationen immer direkt im Sichtfeld, egal, wo er sich gerade im Labor befindet. Für die Navigation innerhalb der digitalen SOPs reichen wenige Kommandos aus, fünf wesentliche sind in Tabelle 1 erläutert. Bei Bedarf kann die Kamera zur Dokumentation oder zum Scannen von QR-Codes verwendet werden, während zu jedem Zeitpunkt beide Hände für die Laborarbeit frei sind.

In Abbildung 1a, b Das weltweit erste Head Mounted Displays (HMD) mit dem Namen „Schwert des Damokles“ (1968). (c) Eines der ersten produktiv genutzten HMDs, das „Integrated Helmet and Display Sighting System“ (IHADSS) für Apache Kampfhubschrauber (1985). (d) Google Glass® (2013) [5–7].

Abbildung 3 sind ausschnittweise die Schritte zur Visualisierung einer digitalen SOP für die Herstellung eines Hefeextrakt-Mediums dargestellt, wie es für die Kultivierung von Bakterien eingesetzt wird. Zu Beginn identifiziert sich der Mensch über einen QR-Code im System. Im nächsten Schritt wird ein Becherglas auf der Waage platziert. Sobald der Mensch durch das „Confirm“ Sprachkommando bestätigt, dass das Becherglas auf der Waage steht, wird diese automatisch tariert. Im nächsten Schritt soll ein Hefeextrakt-Behälter geholt und über seinen QR-Code im System registriert werden. Bei dieser Registrierung werden alle Informationen, die zu dieser Chemikalie bekannt sind, automatisch in der Datenbank abgelegt, zum Beispiel Chemikalienname, Ablaufdatum und Chargennummer. Wird der falsche QR-Code gescannt, bekommt der Mensch eine Fehlermeldung und muss erneut scannen. Die SOP wird erst fortgesetzt, wenn ein korrekter QR-Code gescannt wurde. Im nächsten Schritt wird der Hefeextrakt eingewogen, dabei wird sowohl der Zielwert als auch der aktuelle Wägewert live eingeblendet. Sobald der aktuelle Wägewert innerhalb einer vordefinierten Toleranz um den Zielwert liegt, gilt die Einwaage als erfolgreich und es wird automatisch zum nächsten Schritt gewechselt. Die tatsächlich eingewogene Masse des Hefeextraktes wird automatisch vom System dokumentiert. Nun wird der Hefeextrakt in eine Schraubflasche (Schott-Flasche) überführt. Dieser Ablauf (Abbildung 1a, b) Das weltweit erste Head Mounted Displays (HMD) mit dem Namen „Schwert des Damokles“ (1968). (c) Eines der ersten produktiv genutzten HMDs, das „Integrated Helmet and Display Sighting System“ (IHADSS) für Apache Kampfhubschrauber (1985). (d) Google Glass® (2013) [5–7].

TAB. 1 | BENÖTIGTE KOMMANDOS FÜR DIE DURCHFÜHRUNG EINER DIGITALEN SOP

Bestätigen	Bestätigung, dass eine Aufgabe durchgeführt wurde
Zurück	Geht einen Schritt zurück in der SOP
Wiederhole	Wiederholt einen Schritt in der SOP (z. B., wenn eine Messung nicht richtig funktioniert hat)
Überspringen	Überspringt einen Schritt in der SOP (z. B., wenn vorher Schritte zurück-gegangen wurden)
Scannen	Startet die QR-Code-Scanfunktion der Smartglasses

Abbildung 3, Schritt 2–5) wird für alle anderen Medienbestandteile wiederholt, bis alle Komponenten des Mediums vereinigt sind. Im letzten Schritt wird das fertige Medium mit einem QR-Code-Label versehen, über das auf alle wichtigen Informationen zugegriffen werden kann. Mit dem Scan des Labels ist das hergestellte Medium im System registriert und die SOP beendet.

In einer Mini-Studie wurde untersucht, ob solche digitalen SOPs zu einer Effizienzsteigerung führen können. Dafür haben die Teilnehmer eine MEBAK-SOP für eine photometrische Untersuchung von Bier durchgeführt. Alle Teilnehmer haben die SOP zum ersten Mal durchgeführt. Die eine Hälfte verwendete die papierbasierte SOP und die andere Hälfte die digitale SOP. Die Probanden mit der digitalen SOP waren im Mittel knapp 20 % schneller [11]. Ob dieser signifikante Effizienzgewinn allerdings auch bei anderen SOPs gültig ist, sollte in größeren Studien eingehender untersucht werden.

Wissenschaftliche Apps

Ein anderes Einsatzgebiet für Smartglasses im Labor sind wissenschaftliche Apps. Dabei handelt es sich um kleinere spezialisierte Anwendungen, die keine oder nur eine simple digitale Infrastruktur erfordern.

Ein Beispiel dafür ist das Zählen von koloniebildenden Einheiten auf Agarplatten. Dieses ist in der Wissenschaft und der Qualitätssicherung ein weit verbreitetes Verfahren,

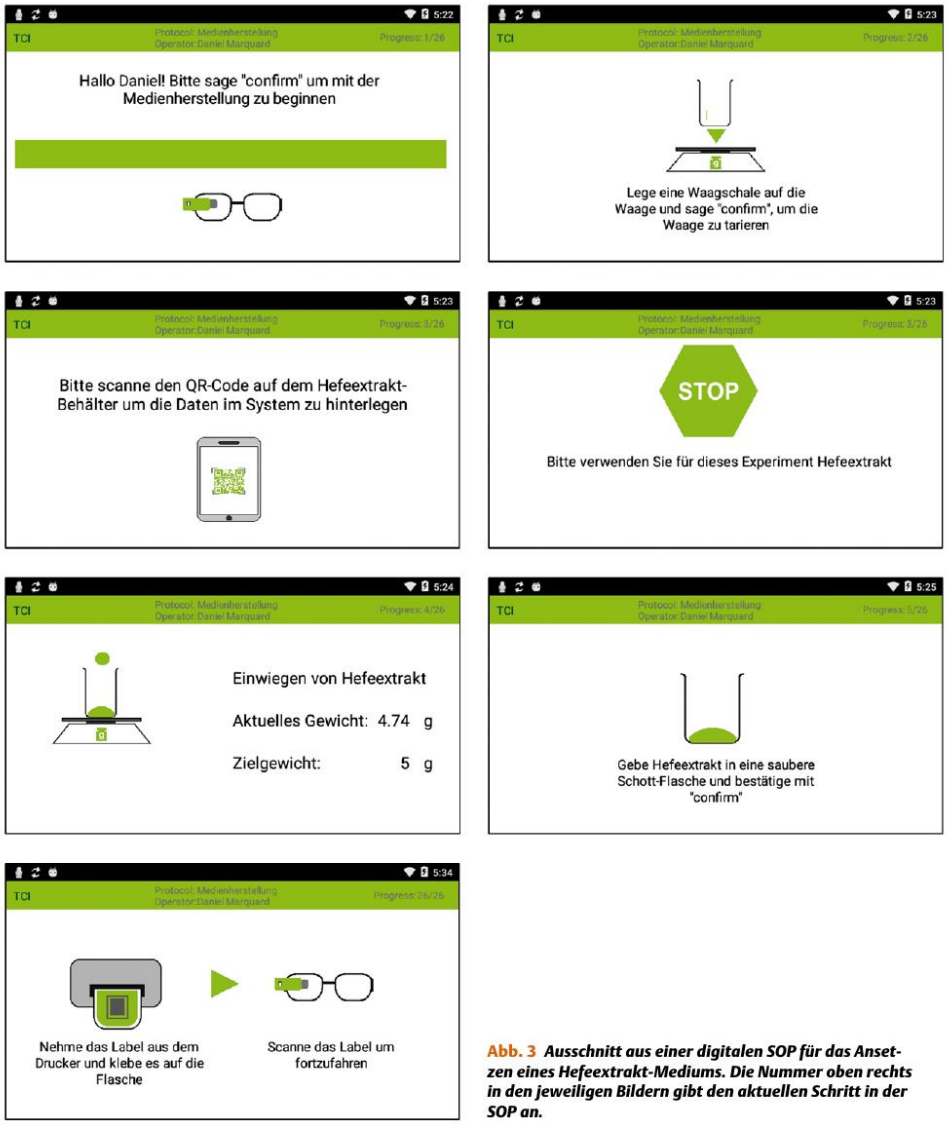
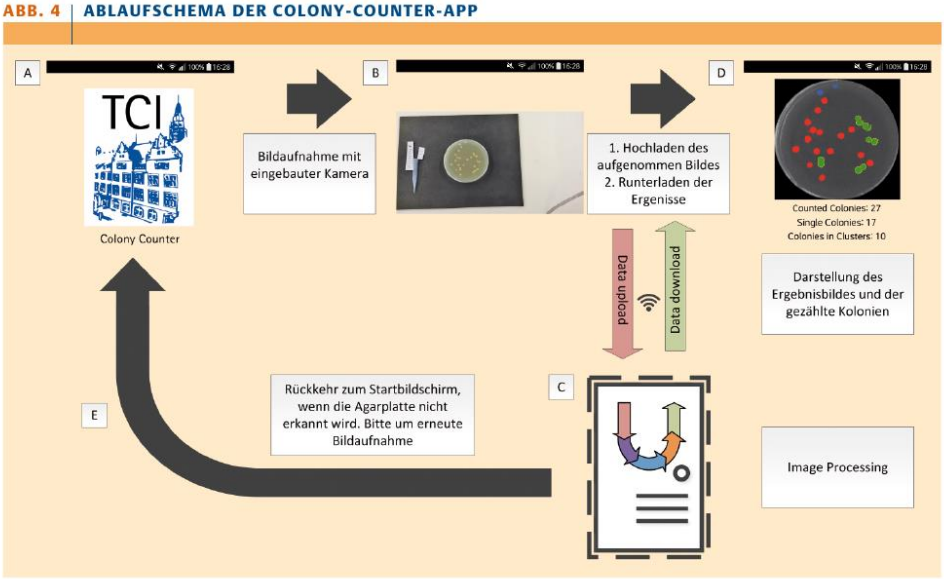


Abb. 3 ***Ausschnitt aus einer digitalen SOP für das Ansetzen eines Hefeextrakt-Mediums. Die Nummer oben rechts in den jeweiligen Bildern gibt den aktuellen Schritt in der SOP an.***

um die Anzahl lebender Mikroorganismen in einer Probe zu bestimmen. Es gibt Geräte, die die Agarplatten automatisch auszählen können, diese sind aber meist nur in spezialisierten Laboren zu finden. Besonders im akademischen Bereich werden die Agarplatten meist händisch ausgezählt. Dies kann mehrere Minuten pro Platte in Anspruch nehmen und ist fehleranfällig, da zur Zeitersparnis häufig nur ein Teil der Platte ausgezählt und dann auf die Gesamtplatte hochgerechnet wird. Eine Alternative bietet die Colony-Counter-App, welche anhand von Bildern die Anzahl von koloniebildenden Einheiten auf Agarplatten bestimmt [13]. Die Colony-Counter-App verwendet dafür eine Server-Client-Architektur. Hierbei wird die Kamera der Smartglasses verwendet, um ein Bild aufzunehmen. Dieses wird aber nicht in der Brille selbst verarbeitet, sondern an einen Server übertragen und dort mit einem Bildverarbeitungsalgorithmus

ABB. 4 | ABLAUFSCHEMA DER COLONY-COUNTER-APP



(A) Begrüßungsbildschirm. (B) Vorschau des aufgenommenen Bildes. (C) Daten-Upload und Bildverarbeitung durch server-basierten Algorithmus und Download der Ergebnisse. (D) Anzeige der gezählten Kolonien und des verarbeiteten Bildes. (E) Rückkehr zum Startbildschirm, wenn das aufgenommene Bild nicht auswertbar war.

ausgewertet (Abbildung 4). Die Ergebnisse der Auswertung werden nun an die Smartglasses zurückübertragen und ins Sichtfeld eingeblendet. Im Anschluss kann dann entschieden werden, ob das Ergebnis der Auswertung zufriedenstellend ist, ansonsten kann ein neues Bild aufgenommen und die Auswertung erneut durchgeführt werden. Der große Vorteil einer solchen Server-Client-Architektur ist, dass besonders rechenintensive Aufgaben wie das Auswerten von Bildern von einem leistungsstarken Server übernommen werden können. Dies beschleunigt die Auswertung der aufgenommenen Bilder enorm und schont den Akku der Smartglasses.

Ähnliche Konzepte wurden zur Quantifizierung von Chlorophyll in Blättern von Pflanzen und zur Auswertung von immunochromatographischen Assays verwendet, hierfür wurde in beiden Fällen die Google Glass® verwendet [14, 15].

Ein weiteres Anwendungsbeispiel für wissenschaftliche Apps findet sich bei Toxizitätsstudien von Chemikalien auf Pflanzen, sogenannten Ökotoxikologie-Studien. In einer solchen Studie werden unterschiedliche Konzentrationen von Chemikalien in die Pflanzerde gemischt. Im Anschluss wird über einen bestimmten Zeitraum beobachtet, ob die Chemikalien eine Auswirkung auf das Wachstum und die Physiologie der Pflanze haben. In regelmäßigen Abständen werden alle Pflanzen durch einen Menschen auf definierte Kriterien untersucht.

Bisher wurden die Befunde für die jeweilige Pflanze in ein Formblatt eingetragen. Dieses Vorgehen hat aber zwei entscheidende Nachteile: Zum einem liegen die Informationen nur auf Papier vor und müssen für die statistische Auswertung noch digitalisiert werden und zum anderen ist es für die Untersuchung und visuelle Bewertung der Pflanzen von Vorteil, beide Hände zur Verfügung zu haben. Dann kann die Pflanze mit einer Hand gehalten und mit der anderen Hand durch das Blattwerk gestrichen werden, um nach etwaigen Einflussspuren der Chemikalien wie Wuchsschäden zu suchen. Dies ist schwierig, wenn zusätzlich auch noch Stift und Klemmbrett gehalten werden müssen, so dass eine Person allein dies kaum effektiv durchführen kann. Aus diesem Grund wurde die Smartglasses-Anwendung Plant-Classifier entwickelt. Diese kommt ohne weitere digitale Infrastruktur aus und kann so ortsunabhängig eingesetzt werden.

Ein typischer Einsatz des Plant-Classifiers läuft folgendermaßen ab: Der Mensch setzt sich die Brille auf und registriert sich über einen QR-Code in der Anwendung (Abbildung 5). Dadurch werden alle Messungen der Person zugeordnet. Nun nimmt der Mensch die erste Pflanze und scannt den zugehörigen QR-Code auf dem Topf. Im QR-Code sind alle relevanten Informationen hinterlegt, etwa Studienzugehörigkeit, Pflanzenart oder zu untersuchende Merkmale. Dadurch wird von der Anwendung automatisch die korrekte Maske für die Befundung geladen. Der Mensch

kann über Sprachkommandos die jeweiligen Merkmale in der Maske bewerten, Beispiele für Bewertungsstufen wären: „normal“, „leicht“, „mittel“ und „hoch“. Jede Eintragung wird auf dem Display der Smartglasses direkt visualisiert. Für die Eingabe gibt es auch Komfortfunktionen, die es erlauben, alle verbleibenden Merkmale auf einen Wert zu setzen. Dies beschleunigt die Aufnahme der Merkmale und ist beispielsweise hilfreich, wenn alle Merkmale der Pflanze normal sind. Nachdem alle Merkmale aufgenommen wurden, werden die Daten auf den Smartglasses in einer Ergebnisdatei abgespeichert. Nun kann der QR-Code der nächsten Pflanze gescannt werden und der Prozess wird wiederholt, bis alle Pflanzen untersucht wurden. Im Anschluss kann die Smartglasses an einen Computer angeschlossen und die Ergebnisdatei für die statistische Auswertung übertragen werden.

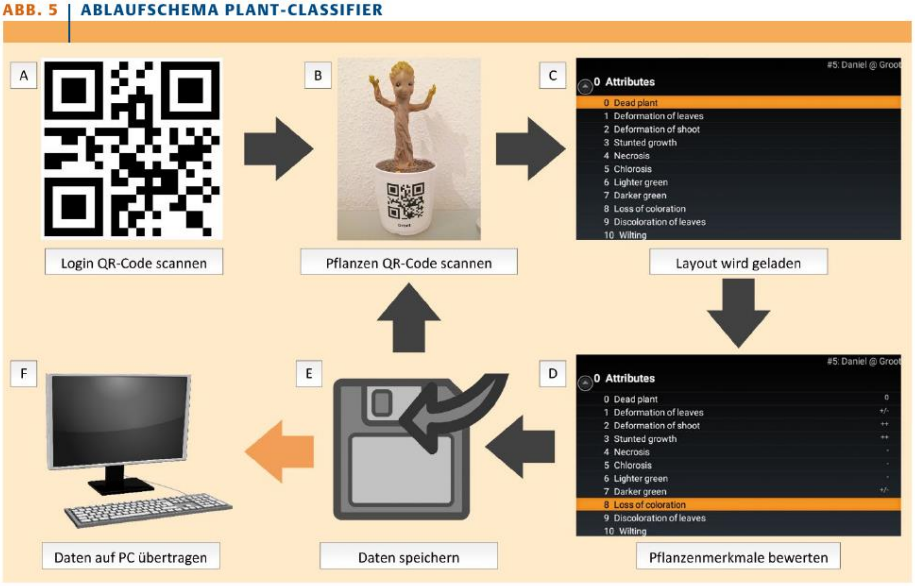
Die Plant-Classifier-Anwendung auf den Smartglasses bietet für den Menschen bei der Untersuchung einige Vorteile. Er hat für die Untersuchung der Pflanzen immer beide Hände frei und trotzdem sind alle Eingaben der aktuellen Pflanze immer im Sichtfeld eingeblendet. Über die QR-Codes werden die Pflanzen sicher identifiziert und Verwechslungen sind ausgeschlossen. Die generierten Daten liegen maschinenlesbar vor und können einfach in die Auswertung überführt werden. Fehler, die beim Übertragen von Papier in den Computer entstehen, sind ausgeschlossen.

Fernwartung – Remote Maintenance

In vielen Laboren werden komplexe und hoch spezialisierte Geräte eingesetzt. Die Einweisungen und Wartungen für diese Geräte sind häufig mit sehr hohen Kosten verbunden, da Spezialisten oft erst vom anderen Ende der Welt anreisen müssen. Besonders ärgerlich ist dies, wenn es sich um ein kleines Problem handelt, welches sich durch ein paar einfache Handgriffe lösen ließe, sofern die Menschen vor Ort gewusst hätten wie. Aus diesem Grund gibt es immer mehr Bestrebungen zur Remote Maintenance, also zur Fernwartung. Bei dieser sollen die Mitarbeiter bei der Behebung von Problemen digital angeleitet werden und so das Problem lösen, ohne dass ein Techniker anreisen muss.

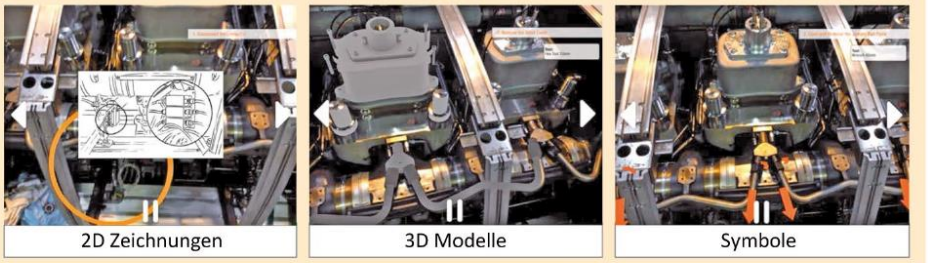
Die einfachste Möglichkeit ist es, beispielsweise Bild-, Videoanaleitungen oder Augmented-Reality-Inhalte für häufig auftretende Probleme zur Verfügung zu stellen. Hier bieten sich Tablets und Notebooks als Medium an [16]. Sobald dies nicht mehr ausreicht, kann ein Techniker mittels Videokonferenz hinzugeschaltet werden. Auch hier bietet sich wieder ein sehr gutes Einsatzgebiet für Smartglasses (Abbildung 6). Über das Mikrofon und die Lautsprecher der Smartglasses können beide Seiten miteinander kommunizieren. Über die Smartglasses-Kamera wird ein Videostream zum Techniker aufgebaut. Dieser zeigt dem Techniker genau das, was der Mensch am Gerät auch sieht (Personal View). Dieser Videostream wird auch auf dem

ABB. 5 | ABLAUFSCHEMA PLANT-CLASSIFIER



(A) Scan des QR-Codes, um sich als Mensch im System anzumelden. (B) Scan eines Pflanzen-QR-Codes. (C): Das für die Pflanze und Studie passende Bewertungslayout wird automatisch geladen. (D) Pflanzenmerkmale werden über Sprachkommandos bewertet; (E) Daten werden in einer Datei auf den Smartglasses gespeichert. Ablauf von B bis E wird solange wiederholt, bis alle Pflanzen untersucht wurden. (F) Export der gesammelten Daten auf einen PC.

ABB. 6 | HILFE BEI DER FERNWARTUNG



Arten der Hilfestellung bei der Fernwartung am Beispiel einer Industriemaschinenwartung inklusive 2D-Zeichnungen und Hervorhebung von relevanten Bildbereichen, 3D-Modellen mit konkreter Positionierung sowie Symbolen, die Handlungsanweisungen verdeutlichen *(modifiziert nach [17])*.

Display der Smartglasses eingeblendet. Je nach Art der verwendeten Software kann der Techniker Markierungen oder 3D-Elemente in den Videostream einblenden, um den Menschen bei der Wartung besser anzuleiten [17]. Die Vorteile bei der Verwendung von Smartglasses sind hier, dass die Videoperspektive dem Sichtfeld des Menschen vor Ort entspricht, dass der Mitarbeiter vor Ort alle Informationen vom Techniker direkt im Sichtfeld hat und vor allem, dass er beide Hände für die eigentliche Wartungsarbeit frei hat.

Der Bereich Fernwartung (Remote Maintenance) ist aufgrund des großen finanziellen Interesses bereits sehr weit entwickelt. Es gibt inzwischen verschiedene Firmen, die Services und Technologien unter Verwendung von Smartglasses anbieten, zum Beispiel die deutsche Firma Oculavis [18].

Herausforderungen beim Einsatz von Smartglasses

Wenn Smartglasses so ein großes Potential für den Einsatz im Labor haben, warum sind sie trotzdem bisher in kaum einem Labor zu finden? Smartglasses sind eine verhältnismäßig neue Technologie, mit der es noch wenig Erfahrungswerte gibt, so müssen rechtliche Aspekte geklärt, technische Limitierungen überwunden und die potenziellen gesundheitlichen Auswirkungen auf den Menschen berücksichtigt werden.

Smartglasses bieten in Kombination mit einer entsprechenden digitalen Infrastruktur viele Vorteile. Dies zeigt sich besonders in regulierten Umfeldern, die nach GMP (Good Manufacturing Practice) oder GLP (Good Laboratory Practice) arbeiten, wie in der Pharmaindustrie. Hier kann die aufwändige Dokumentation vom digitalen System übernommen werden. Damit aber eine neue Technologie wie die Smartglasses in solch einem Umfeld eingesetzt werden kann, muss diese dort erst validiert werden, was je nach Komplexität der Anwendung sehr aufwändig und damit auch kostenintensiv sein kann. Ein weiteres rechtliches Problem ist die potenzielle Möglichkeit, Mitarbeiter zu überwachen. Genau wie in Smartphones sind in fast allen Smartglasses auch Richtungs- und Beschleunigungssensoren verbaut, anhand derer es prinzipiell möglich ist, zu

WAS MAN WISSEN SOLLTE

Smartglasses ermöglichen es, digitale Informationen immer im Sichtfeld und trotzdem beide Hände für die eigentliche Arbeit frei zu haben.

Smartglasses sind Schnittstellen zur digitalen Infrastruktur und können bei stark repetitiven Arbeiten wie der Dokumentation helfen.

Durch Smartglasses können Arbeitsabläufe bei gesteigerter Datenqualität einfacher, sicherer und reproduzierbarer gemacht werden.

Smartglasses haben großes Potential für den Einsatz im Labor, werden aktuell aber nur für spezifische kurze Aufgaben eingesetzt.

Bevor Smartglasses die klassische Schutzbrille im dauerhaften Einsatz ersetzen können, müssen noch rechtliche und medizinische Fragestellungen geklärt werden.

Smartglasses werden stetig weiterentwickelt und könnten zu einer der herausragenden Technologien im Labor der Zukunft werden.

TAKE-HOME MESSAGE

Smartglasses allow digital information to be always within view while using both hands for the actual work.

Smartglasses are an interface to the digital infrastructure and can be used to carry out highly repetitive tasks such as documentation.

Smartglasses can make workflows simpler, safer and more reproducible, while increasing data quality.

Before Smartglasses can replace the classic safety goggles legal and medical issues must be addressed.

Smartglasses offer great potential for use in the laboratory but are currently only useable for specific short tasks.

Smartglasses are constantly evolving and could become one of the key technologies in the laboratory of the future.

bestimmen, was eine Person gerade tut [19]. Dieses Problem verstärkt sich nochmal bei Smartglasses, die als Schutzbrillen getragen werden, da sie im Gegensatz zum Smartphone nicht abgelegt werden dürfen und so der Mitarbeiter keine Möglichkeit hat, seine Privatsphäre aktiv zu schützen. Ähnlich verhält es sich mit der Kamera. Diese zeigt bei Smartglasses im Normalfall die Sicht des Mitarbeiters und kann so ebenfalls sehr effizient zur Überwachung missbraucht werden. Auch die Privatsphäre anderer Personen kann verletzt werden, indem Fotos oder Videos mit den Smartglasses gemacht werden [20]. Deshalb sollten vor der Einführung der Smartglasses am Arbeitsplatz solche Bedenken angesprochen und nach Möglichkeit technisch ausgeschlossen werden, insbesondere um die Akzeptanz der Mitarbeiter für die neue Technologie zu erhöhen.

Eine weitere Limitierung stellen häufig die Smartglasses selbst dar. In vielen Fällen werden Smartglasses mit veralteten Betriebssystemen ausgeliefert. So wurde die Vuzix Blade bei der Veröffentlichung 2019 mit Android 5 ausgeliefert, welches aus dem Jahr 2014 stammt und damit 4 Generationen hinter dem zu dem Zeitpunkt aktuellen Android 9 lag. Dies macht die Entwicklung von Software aufwändiger, da häufig veraltete Softwarebibliotheken verwendet werden müssen. Aufgrund der häufig geringen Rechenleistung der Smartglasses, müssen rechenintensive Aufgaben an Server ausgelagert werden. Dies hat zur Folge, dass eine komplexere Infrastruktur benötigt wird, die überall verfügbar sein muss, wo die entsprechende Anwendung eingesetzt werden soll. Bei vielen Smartglasses sind die Kapazitäten der internen Akkus sehr gering und es werden Zusatzakkus benötigt. Diese können an der Schutzbrille selbst befestigt werden, was das Gewicht erhöht. Alternativ können die Zusatzakkus am Körper getragen werden, dies kann jedoch die Bewegungsfreiheit des Kopfes einschränken. Gerade bei älteren Smartglasses-Modellen war die Bildschirmauflösung gering, was die Darstellung von komplexen Inhalten schwierig machte.

Der dritte wichtige Aspekt ist die Frage, ob die Verwendung von Smartglasses eine gesundheitliche Belastung für den Mitarbeiter darstellt. Das zusätzliche Gewicht, das

2: DIGITALE STANDARD-ARBEITSABLÄUFE ERSTELLEN

Um digitale Standard-Arbeitsabläufe, Standard Operation Procedures (SOPs), durchführen zu können, muss eine digitale Infrastruktur im Labor vorhanden sein, welche die Laborgeräte ansteuert und die Dokumentation der Ergebnisse übernimmt. Wie eine solche Infrastruktur aussehen könnte, kann bei Porr et al. und Austerjost et al. nachgelesen werden [10, 11]. Ist eine solche Infrastruktur vorhanden, können digitale SOPs erstellt werden. Dies geschieht im Wesentlichen in vier Schritten (Abb. Info 2-1).

Im ersten Schritt muss der abzubildende Gesamtablauf in kleine einzelne Arbeitsschritte zerlegt werden. Die Zerlegung muss hierbei bis auf einzelne Handgriffe erfolgen, was nochmal deutlich detaillierter ist als bei papierbasierten SOPs. Beispielsweise würde die papierbasierte Anweisung „Wiege 5 g Glucose in ein Becherglas ein" in einer digitalen SOP in die folgenden Schritte aufgeteilt werden: „Platziere Becherglas auf Waage", „Tariere Waage" und „Gebe 5 g Glucose in ein Becherglas". Nicht alle Schritte bekommt der Mensch im Labor angezeigt, aber sie müssen dennoch erfasst werden. So würde der Mensch bestätigen, dass das Becherglas auf der Waage steht, der Tarierschritt würde dann jedoch automatisch ausgeführt und vor dem Menschen versteckt werden.

Die Einzelschritte müssen als nächstes in ein maschinenlesbares Format gebracht werden. Hierfür werden häufig Markup Languages wie „YAML Ain't Markup Language" (YAML) und Extensible Markup zuLanguage (XML), oder Datenformate wie JavaScript Object Notation (JSON) verwendet. Ein alternativer Ansatz ist es, diese direkt in einer vollwertigen Programmiersprache wie Python oder C# abzubilden, was mehr Freiheiten erlaubt, aber auch anspruchsvoller in der Erstellung ist. Alle diese Umsetzungen sind für den Menschen lesbar, was ein manuelles Erstellen und Bearbeiten ermöglicht. Allerdings ist es immer sinnvoll, so viel wie möglich von der Erstellung des maschinenlesbaren Formats zu automatisieren, vor allem um Schreib- und Syntaxfehler zu vermeiden.

Im dritten Schritt muss eine geeignete und intuitiv verständliche Visualisierung für die manuellen Schritte gefunden werden. Dafür bietet es sich an die Informationen mit Piktogrammen zu verdeutlichen und so wenig Text wie möglich zu verwenden. Abbildung 3, besonders wenn die Informationen auf Smartglasses angezeigt werden sollen. Hierbei sollte ebenfalls so viel wie möglich automatisiert werden, dies kann zum Beispiel durch eine Standardisierung des Layouts geschehen.

Der letzte Schritt ist die Validierung der SOP. Dies findet auch bei papierbasierten SOPs statt, gestaltet sich bei digitalen SOPs aber aufwändiger, da es mehr Fehlerquellen gibt. So können Schreibfehler in der maschinenlesbaren Datei zu Problemen führen. Außerdem muss geprüft werden, ob alle Geräte wie erwartet angesteuert werden. Darüber hinaus muss sichergestellt sein, dass die Visualisierung verständlich und eindeutig ist. Sollten Probleme bei der Testung entdeckt werden, muss in den zum Problem gehörigen Erstellungsschritt zurückgegangen und das Problem behoben werden. Dies kann häufig auch eine Anpassung der Folgeschritte erfordern.

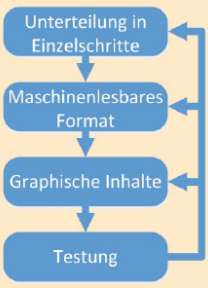


Abb. Info 2-1 ***Fließdiagramm zur Erstellung digitaler SOPs.***

durch die Smartglasses am Kopf getragen wird, kann zu stärkerer Beanspruchung der Hals-Nacken-Schulter-Muskulatur führen. Neben dem Kopf werden auch die Augen durch Smartglasses beansprucht, da das Auge ständig die Sehschärfe zwischen „nah“, für die Informationen auf den Smartglasses, und „fern“ für das reguläre Sehen anpassen muss; hierbei spricht der Mediziner von Akkommodation. Zusätzlich ist das Sichtfeld bei nicht transparenten Displays, wie sie in vielen Smartglasses aktuell verbaut sind, immer eingeschränkt, auch wenn keine Informationen darauf angezeigt werden.

Wie stark sich diese Einflüsse auf den Menschen auswirken, wurde von der Bundesanstalt für Arbeitsschutz und Arbeitsmedizin (baua) 2016 in einer Studie untersucht [21]. Hierbei wurde die Arbeitsbelastung von Smartglasses mit der von Tablets gegenübergestellt. In dieser Studie haben die Probanden bis zu 4 Stunden Aufgaben mit Smartglasses oder einem Tablet ausgeführt. Es zeigte sich, dass die Verwendung von Smartglasses keinen Einfluss auf das Sehvermögen der Probanden hatte. Es wurde weder eine Beeinträchtigung der Akkommodation noch des Gesichtsfeldes beobachtet. In der Studie wurde eine geringe Mehrbelastung der Hals-Nacken-Schulter-Muskulatur festgestellt, wobei hier der subjektive Eindruck der Probanden die messbaren Beeinträchtigungen deutlich überstieg. Ebenfalls wurde auch die psychische Belastung untersucht. Diese war am Anfang bei Smartglasses im Vergleich zum Tablet sowohl objektiv als auch subjektiv höher. Nach einer Gewöhnungsphase war die objektive psychische Belastung bei der Nutzung von Smartglasses zwar identisch, aber die gefühlte weiter deutlich höher. Ob eine dauerhafte Nutzung von Smartglasses zu gesundheitlichen Beeinträchtigungen führen kann, muss erst noch untersucht werden.

Anzumerken ist auch, dass die Studie der baua mit den ersten Generationen der kommerziell verfügbaren Smartglasses durchgeführt wurde und es seitdem einige Weiterentwicklungen gab, welche den Tragekomfort und die technische Leistungsfähigkeit erheblich verbesserten. In Abbildung 7 sind beispielhaft die verschiedenen Smartglasses Modelle der Firma Vuzix dargestellt. In jeder Generation von M100 über M300 bis zur M400 wird die Hardware, der Prozessor, der Arbeitsspeicher, die Kamera und das Display besser. Besonders die höheren Displayauflösungen tragen maßgeblich zum besseren Empfinden des Menschen bei. Darüber hinaus gab es auch Designentscheidungen, die den Tragekomfort deutlich erhöhen. Ab dem Modell M300 wurde ein Zusatzakku auf der dem Display gegenüberliegenden Seite verbaut, was das Gesamtgewicht des Systems erhöht, aber durch die ausgeglichene Gewichtsverteilung deutlich angenehmer zu tragen ist. Die Vuzix Blade wirkt auf den ersten Blick wie eine etwas klobige, normale Brille. Das Display ist im Brillenglas integriert, so dass bei inaktivem Display keine Beeinträchtigung des Sichtfeldes vorliegt. Ebenso ist sie mit 90 g Gesamtgewicht relativ leicht. Allerdings sind die Hardwarekomponenten im Vergleich zur fast zeitgleich veröffentlichen M400 relativ schwach. Aber auch dieses Modell ist nach Meinung der Autoren für eine tägliche Anwendung noch nicht geeignet, die Entwicklung geht aber in die richtige Richtung. Neben der Firma Vuzix gibt es auch einige andere Firmen, die spannende Konzepte für Smartglasses entwickeln. Ein Beispiel hierfür ist die amerikanische Firma North. Diese verwendet eine Technologie, bei der die Bildinformationen direkt ins Auge projiziert wird, so dass kein Display mehr notwendig ist [22].

ABB. 7 | GENERATIONEN DER VUZIX SMARTGLASSES

				
Modell	M100	M300	M400	Blade
Prozessor	ARM Cortex-A9 dualcore	Intel Atom dualcore	Qualcomm XR1 octscore	Quad Core ARM A53
RAM	1 GB	2 GB	6 GB	1 GB
Displaytyp/ Displayauflösung	Undurchsichtig/432×240 Pixel	Undurchsichtig/640×360 Pixel	Undurchsichtig/640×360 Pixel	Durchsichtig/480×480 Pixel
Kamera	5 MP Foto; 1080p Video	10 MP Foto; 1080p Video	12 MP Foto; 4k Video	8 MP Foto; 1080p Video
Akkuleistung*	600 mAh	1000 mAh	860 mAh	470 mAh
Betriebssystem	Android 4.01	Android 6	Android 8.1	Android 5.1.1
Gewicht**	84 g	152 g	202 g	90 g
Erscheinungsjahr	2014	2017	2019	2019

***(*) Gesamtakkuleistung inklusive an Brille getragener Zusatzakku, siehe M300 und M400. (**) Gesamtgewicht inklusive Schutzbrille und an der Brille getragener Zusatzakkus** [23, 24]**.**

Fazit

Insgesamt zeigt sich, dass es noch einige rechtliche und medizinische Fragestellungen zu klären gibt und es einiger weiterer technischer Entwicklungen bei den Smartglasses bedarf, bevor diese auf täglicher Basis im Labor eingesetzt werden können. Zurzeit sind Smartglasses daher etwas für Spezialaufgaben wie Remote Maintenance oder für einzelne Experimente wie beim Plant-Classifier, so dass die Smartglasses nur für wenige Stunden und auch nicht jeden Tag zum Einsatz kommen.

Allerdings verbessern sich die Smartglasses mit jeder Generation deutlich und es gibt immer wieder neue Technologien, die das Feld der Smartglasses revolutionieren könnten. Ob Smartglasses die klassische Schutzbrille vollständig ersetzen werden, ist schwer vorherzusagen und hängt stark von der technischen Entwicklung und von der Akzeptanz der Mitarbeiter im Labor ab. Sollten Smartglasses im Verbraucherbereich Verbreitung finden, was mit den Google Glass® schon 2013 einmal vergeblich versucht wurde, wird dies den Vorgang sicherlich beschleunigen. Auch vorstellbar wäre es, dass ein anderes Technologiekonzept die Smartglasses überflüssig macht, etwa eine Kombination aus Smartspeakern und mobilen Bildschirmen. Smartglasses haben eindeutig ein außerordentliches Potential für Anwendungen im Labor und sind auch schon heute ein wertvolles Werkzeug für spezielle Aufgaben.

Zusammenfassung

Schutzbrillen müssen im Labor getragen werden, also warum diesen nicht einen Mehrwert geben? Der Vorteil an Smartglasses ist, dass der Träger digitale Informationen direkt im Sichtfeld eingeblendet bekommt, über Sprachkommandos durch diese navigieren kann und trotzdem beide Hände für die eigentliche Arbeit frei hat. Diese Stärke kommt bei der Durchführung von digitalen Workflows, dem Einsatz von wissenschaftlichen Apps und der Remote Maintenance besonders zum Tragen. Aktuell werden Smartglasses nur für Spezialanwendungen im Labor eingesetzt, die zeitlich begrenzt sind. Bevor ein flächendeckender Einsatz von Smartglasses möglich ist, müssen erst noch rechtliche und medizinische Fragestellungen geklärt werden. Um einen ganztägigen Einsatz im Labor zu ermöglichen, müssen sich Technologie und Tragekomfort der Smartglasses noch verbessern. Insbesondere in den voll digitalisierten Laboren, welche in der Zukunft entstehen werden, können Smartglasses ein sehr wertvolles und mächtiges Werkzeug für die tägliche Laborarbeit werden.

Summary

Protective goggles have to be worn in the laboratory, so why not give them an extra benefit? The advantage of Smartglasses is to display digital information directly in the wearer's field of view and to navigate through them by voice commands while still having both hands free for practical work. This strength is particularly useful when performing digital workflows, using scientific apps and remote maintenance. Currently Smartglasses are only useable for special applications in the laboratory that are limited in time. Before a widespread use of Smartglasses is possible, legal and medical issues must be addressed and solved first. In order to enable all-day use in the laboratory the technology and wearing comfort of Smartglasses still have to be improved. Especially in fully digitalized laboratories upcoming in the future, Smartglasses can become a very valuable and powerful tool for daily laboratory work.

Schlagwörter

Smartglasses, Digitalisierung, Labor der Zukunft, Internet of Things, Scientific Apps, Remote Maintenance, Fernwartung

Literatur

[1] Sutherland, I. E. In: *Proceedings of the December 9–11, 1968, fall joint computer conference, part I on - AFIPS '68 (Fall, part I)*, ACM Press, New York, New York, USA, **1968**, S. 757.
[2] Mann, S. *Computer (Long. Beach. Calif).* **1997**, *30*, 25–32. https://doi.org/10.1109/2.566147
[3] Rash, C. E. und Martin, J. S. *The impact of the US Army's AH-64 helmet-mounted display on future helmet design*,**1988**.
[4] Carmigniani, J. und Furht, B. *Handbook of Augmented Reality*, (Hrsg. Furht, B.) Springer New York, New York, NY, **2011**. 3–47.
[5] Vrroom.buzz. https://vrroom.buzz/vr-news/guide-vr/sword-damocles-1st-head-mounted-display
[6] Wikipedia.org. https://en.wikipedia.org/wiki/Helmet-mounted_display#/media/File:Integrated_Helmet_Display_Sight_System.jpg
[7] Wikipedia.org. https://de.wikipedia.org/wiki/Datei:A_Google_Glass_wearer.jpg
[8] Mitrasinovic, S., Camacho, E., Trivedi, N. *et al. Technol. Heal. Care* **2015**, *23*, 381–401. https://doi.org/10.3233/THC-150910
[9] Farrell, W. A. *Perform. Improv.* **2018**, *57*, 19–28. https://doi.org/10.1002/pfi.21775
[10] Porr, M., Marquard, D., Stanislawski, N. *et al. Chemie-Ingenieur-Technik* **2019**, *91*, 285–293. https://doi.org/10.1002/cite.201800090
[11] Austerjost, J., Bargholz, M., Porr, M. *et al. BrewingScience* **2019**, *72*, 1–9. https://doi.org/10.23763/BrSc18-20austerjost
[12] Austerjost, J., Porr, M., Riedel, N. *et al. SLAS Technol. Transl. Life Sci. Innov.* **2018**, *23*, 476–482. https://doi.org/10.1177/2472630318788040
[13] Austerjost, J., Marquard, D., Raddatz, L. *et al. Eng. Life Sci.* **2017**, *17*, 959–966. https://doi.org/10.1002/elsc.201700056
[14] Cortazar, B., Koydemir, H. C., Tseng, D. *et al. Lab Chip* **2015**, *15*, 1708–1716. https://doi.org/10.1039/C4LC01279H
[15] Feng, S., Caire, R., Cortazar, B. *et al. ACS Nano* **2014**, *8*, 3069–3079. https://doi.org/10.1021/nn500614k
[16] Webel, S., Bockholt, U., Engelke, T. *et al. Rob. Auton. Syst.* **2013**, *61*, 398–403. https://doi.org/10.1016/j.robot.2012.09.013
[17] Aromaa, S., Aaltonen, I., Kaasinen, E. *et al.* In: *Proceedings of the 20th International Academic Mindtrek Conference on – AcademicMindtrek '16*, ACM Press, New York, New York, USA, **2016**, S. 235–242.
[18] Oculavis GmBH. https://oculavis.de
[19] Attal, F., Mohammed, S., Dedabrishvili, M. *et al. Sensors* **2015**, *15*, 31314–31338. https://doi.org/10.3390/s151229858
[20] Kotsios, A. *Int. J. Law Inf. Technol.* **2015**, *23*, 157–185. https://doi.org/10.1093/ijlit/eav003
[21] baua: Bundesanstalt für Arbeitsschutz und Arbeitsmedizin. *Head Mounted Displays – Arbeitshilfen der Zukunft. Bedingungen für den sicheren und ergonomischen Einsatz monokularer Systeme*, **2016**.
[22] North. https://www.bynorth.com/
[23] Vuzix. https://www.vuzix.com/products/compare-vuzix-smart-glasses

[24] Vuzix. http://files.vuzix.com/Content/pdfs/Vuzix-M100-Product-Sheet-01-01-2016.pdf
[25] Wikimedia.org. https://commons.wikimedia.org/wiki/File:Oculus-Rift-CV1-Headset-Front.jpg
[26] Wikimedia.org. https://commons.wikimedia.org/wiki/File:Ramahololens.jpg
[27] Kuhlemann, I., Kleemann, M., Jauer, P. *et al. Healthc. Technol. Lett.* **2017**, *4*, 184–187. https://doi.org/10.1049/htl.2017.0061

Die Autoren

Daniel Marquard, Jahrgang 1988, studierte in Hannover und Cambridge Life Science und erlangte 2013 seinen Masterabschluss an der Leibniz-Universität Hannover. 2017 promovierte er am Institut für Technische Chemie über die Entwicklung von Bildverarbeitungsalgorithmen für biotechnologische Prozesse. Heute ist er Postdoc und forscht im Bereich Labordigitalisierung an standardisierter Gerätekommunikation, Datenmanagement, digitalen Workflows und Interaktionsmedien für das digitalisierte Labor.

Marc Porr, Jahrgang 1992, studierte in Hannover Life Science und erlangte 2017 seinen Masterabschluss an der Leibniz-Universität Hannover. Seitdem promoviert er am Institut für Technische Chemie der Leibniz Universität Hannover im Bereich Labordigitalisierung. Neben der Entwicklung und Umsetzung von Systemarchitekturen forscht er an standardisierter Gerätekommunikation, Datenmanagementsystemen, digitalen Workflows und Nutzerinteraktionsmedien für das digitalisierte Labor.

Ferdinand Lange, Jahrgang 1990, studierte in Hannover Informatik und erlangte 2018 seinen Masterabschluss an der Leibniz-Universität Hannover. Seitdem ist er wissenschaftlicher Mitarbeiter am Institut für Technische Chemie der Leibniz-Universität Hannover und ist hier federführend bei der Entwicklung und Implementierung von Softwarearchitekturen für Labordigitalisierungsvorhaben beteiligt.

Jonas Austerjost, Jahrgang 1989, studierte in Hannover und Stanford Life Science und erlangte 2014 seinen Masterabschluss an der Leibniz-Universität Hannover. Im Anschluss arbeitete er als wissenschaftlicher Mitarbeiter am Institut für Technische Chemie der Leibniz-Universität Hannover und promovierte dort 2019 über die Entwicklung innovativer Automationslösungen für die chemische und biotechnologische Laborumgebung. Heute arbeitet er als Wissenschaftler in der Corporate Research Division der Sartorius AG in Göttingen und beschäftigt sich mit der Entwicklung von digitalen Tools für die Unterstützung und Optimierung von Laborworkflows.

Sascha Beutel, Jahrgang 1971, studierte Chemie an der Leibniz-Universität Hannover. Er promovierte dort im Jahr 2000 am Institut für Technische Chemie. Zudem habilitierte er sich im Jahr 2018 im Fachgebiet Technische Chemie. Er leitet eine eigene Forschungsgruppe und forscht im Bereich Bioprozesstechnik zur Kultivierung von Prokaryoten, zu enzymtechnologischen Verfahren, Methoden zur Aufarbeitung biotechnologischer Produkte und an neuen Sensorkomponenten sowie daran, neue Technologien wie 3D-Druck, Datenbrillen oder Sprachinteraktion für das Labor zu adaptieren.

Korrespondenzadresse:
PD Dr. Sascha Beutel
Leibniz-Universität Hannover
Institut für Technische Chemie
Callinstraße 5
D-30167 Hannover
E-Mail: Beutel@iftc.uni-hannover.de

3.5 EINBINDUNG VON SMARTGLASSES ALS DIGITALE UNTERSTÜTZUNG BEI STANDARDISIERTEN ARBEITSABLÄUFEN

Eine besondere Herausforderung bei der Übertragung analoger Arbeitsabläufe in das digitalisierte Umfeld stellt die Formulierung von standardisierten und maschinen-verständlichen Ablaufbeschreibungen dar. Traditionell liegen SOPs in unstrukturierter und nicht standardisierter Form vor. Für den Menschen sind solche Beschreibungen – meistens – aufgrund seines ausgeprägten Abstraktionsvermögens verständlich und umsetzbar. Für die Umsetzung im digitalisierten Labor müssen jedoch Wege gefunden werden, diese Abläufe zwar weiterhin für den Menschen gut erstell- und änderbar zu halten, sie jedoch gleichzeitig so zu strukturieren, dass sie maschinenlesbar sind (siehe Abbildung 25).

Im folgenden Artikel wird eine Architektur vorgestellt, die das Formulieren und Ausführen von SOPs mit digitaler Unterstützung erlaubt. Als Nutzerinteraktionsmedium werden dabei SmartGlasses eingesetzt, die dem Nutzer den aktuellen Arbeitsschritt visuell präsentieren und ihm eine Navigation durch die Schritte der SOP erlauben. Die Anbindung von Messgeräten ist ebenfalls möglich. Die Architektur wird beispielhaft mit Standard-Messmethoden der brautechnischen Analyse gezeigt und zur Vervollständigung wurde eine empirische Studie zu Akzeptanz und Nutzen der digitalen Unterstützung mit laborerfahrenen Mitarbeitern durchgeführt.

Der Fokus liegt auf der standardisierten und maschinenlesbaren Formulierung von Protokollen. Dazu werden YAML (*YAML Ain't Markup Language*) Dateien zur Ablaufbeschreibung geschrieben. Diese sind durch ihre textbasierte und klarnamenorientierte Syntax für den Menschen leicht verständlich und erlauben gleichzeitig eine präzise maschinelle Interpretation. Die Verwendung einer textuellen Beschreibungssprache erlaubt eine intuitive Semantik der Protokollbeschreibungssprache. Gleichzeitig können über einen Plugin-Mechanismus einfache Berechnungen integriert werden.

In dieser Arbeit werden wichtige Prinzipen zur standardisierten Formulierung von Abläufen für das digitalisierte Labor gezeigt. Diese wurden im Laufe der Entwicklung des in dieser Arbeit gezeigten Konzepts weiter verfeinert und vervollständigt.

ABBILDUNG 25: GRAPHICAL ABSTRACT DES ARTIKELS: „A FLEXIBLE IT INFRASTRUCTURE FOR THE INTEGRATION OF SMARTGLASSES INTO THE BREWING LABORATORY AS A DIGITAL SUPPORT FOR STANDARD ANALYSIS WORKFLOWS".

1 January / February 2019 (Vol. 72) BrewingScience

J. Austerjost, M. Bargholz, M. Porr, D. Geier, T. Becker, D. Marquard, P. Lindner, T. Scheper and S. Beutel

A flexible IT infrastructure for the integration of smartglasses into the brewing laboratory as a digital support for standard analysis workflows

Standard operating procedures (SOPs) are an often-used medium for the reproducible step-by-step execution of complex laboratory operations. Even in today's era of digitalization, most SOPs are still paper-based. This might cause disorder within the lab and often distracts the experimenter from the actual experiment. Furthermore, most of the experimental documentation nowadays is still done manually by transferring experimental data and results in paper-based laboratory journals, which is quite sensitive to errors as well as time-consuming. We developed a digital laboratory infrastructure that enables interactive hands-free experiment guidance and instrument control via smart safety goggles, also called smartglasses. Instructions for an SOP and experimental data can be displayed directly into the experimenter's field of view and triggered by voice commands. Experimental data and working steps can be processed and documented instantly. The developed laboratory infrastructure allows for the flexible integration of laboratory instruments and SOPs. Different SOPs commonly used for the spectrophotometric analysis of beer (MEBAK methods) were implemented showing the applicability of the established system in the brewing laboratory. Furthermore, a benchmark study of the system was performed. The developed solution enables a faster experiment execution and analysis than conventional paper-based SOPs and is a first step towards the integration of smart safety goggles for the support of laboratory workflows. This might pave the way to easier process documentation and an increase of laboratory workflow efficiency.

Descriptors: smartglasses, laboratory automation, wearables, assisted reality

1 Introduction

More than 50 years have passed since the first head-mounted displays (HMDs) were introduced. Pioneers like Ivan Sutherland were the first that conceived and technically implemented the idea of a head-worn device that displays graphical content into the user's field of view in the late 1960s [1]. The device created by Sutherland and his colleagues was called "The Sword of Damocles", whose name was inspired by the necessity to mount it to the ceiling due to its massive weight and its bulky head tracking mechanism. Miniaturization of electronics and optics have evolved during the decades since, and so have head-mounted displays [2]. Smartglasses size has ranged from backpack solutions in the 1980s to smartglasses that now nearly have the dimensions of regular glasses [3].

After Google rolled out their smartglasses model "Google Glass" in 2013, smartglasses technology became available to a broader audience. Other manufacturers followed Google's example and released models with similar technical characteristics. Although most of the current smartglasses models were not successful in the consumer field, many applications emerged within professional areas. Applications of smartglasses have been evaluated in healthcare, logistics, and manufacturing as a documentation respectively commissioning tool or as a support during workflows [4–7].

Recently, the idea of giving tasks in the laboratory a digital surplus has caught the interest of researchers in different scientific disciplines. Integration efforts extend beyond smartglasses to virtual assistants and other smart devices such as tablets and smartphones that are subject to current research in which their potential as a supporting tool in laboratory settings is evaluated [8, 9]. Smartglasses provide crucial advantages over handheld devices. These advantages include hands-free operation by voice commands and the direct display of information in the user's field of

https://doi.org/10.23763/BrSc18-20austerjost

Authors

Jonas Austerjost, Institute of Technical Chemistry, Leibniz University Hannover, Hannover, Germany, Institute of Brewing and Beverage Technology, Technical University of Munich, Freising, Germany; Malte Bargholz, Marc Porr, Daniel Marquard, Patrick Lindner, Thomas Scheper, Sascha Beutel, Institute of Technical Chemistry, Leibniz University Hannover, Hannover, Germany; Dominik Geier, Thomas Becker, Institute of Brewing and Beverage Technology, Technical University of Munich, Freising, Germany; corresponding author: beutel@iftc.uni-hannover.de

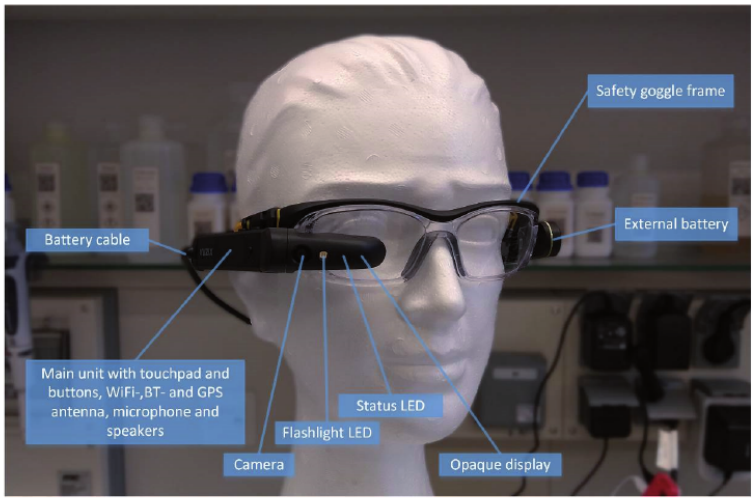


Fig. 1 Smart safety goggles used for the evaluation of the developed digital laboratory architecture (Smartglasses Vuzix M300, Vuzix Corp., USA)

view. Furthermore, safety goggles are mandatory in most laboratories – so why not give them an additional digital benefit? A handful of researchers already seized the idea of bringing smartglasses into the laboratory.

Cortazar et al. used a Google Glass device for the quantification of chlorophyll in plant leaves, while *Feng* et al. developed a Google Glass application for the processing of immunochromatographic assays [10, 11]. Both applications take advantage of a client-server model, where the internal smartglasses camera captures images. Subsequently, the image data are transmitted to an external server for processing. The results of the processing can then be retrieved by the smartglasses and shown in the user's field of view. This approach has the advantage of conserving the battery and processing power of the smartglasses by sourcing out the hardware demanding processing task to a more powerful server unit. A similar concept was worked out by *Austerjost* et al. who used smartglasses and smartphones for the determination of colony-forming units on agar plates [12]. Other applications within the laboratory include the use of smartglasses in physics education as a tool for better experiment perception [13].

The use of smart safety goggles within the laboratory offers the potential to revolutionize experimental work. Not only can built-in sensors (e.g. microphone and camera) be used to collect experimental data, but also external sensors or instruments can be connected using the internal Bluetooth or WiFi capabilities (see Fig. 1). We developed a digital laboratory architecture that allows the easy creation of standard operating procedures (SOPs) that can be retrieved and displayed by smartglasses that act as the front-end. These SOPs can be triggered and navigated hands-free using voice commands. Furthermore, laboratory instruments can be integrated into the back-end architecture, allowing the control of these devices and the display of instrument data into the experimenter's field of view. Measured experimental data and workflow data are saved to a database. Therefore, experimental results can be traced back to their source at a later date. The evaluation of this setup has been carried out using different SPOs covering the spectrophotometric analysis of beer [14]. A benchmark of the developed system was performed by comparing the execution time of a paper-based SOP with the execution time of a smartglasses-based SOP.

2 Materials and methods

2.1 Back-end application

The developed back-end application is a JavaScript application running on NodeJS v8, a cross-platform native JavaScript interpreter (https://nodejs.org/en/). It provides both a JSON/HTTP API (Application Programming Interface) for retrieving and creating protocols and a separate WebSocket API that manages all instrument measurements that are triggered via the backend.

The JSON API stack consists of the express HTTP server (http://expressjs.com/de/), a database backend provided by the popular ORM (Object Relational Mapper) sequelize.js (http://docs.sequelizejs.com/) and custom controller code, which translates HTTP requests to database actions. Database access is also shared with the WebSocket stack, which is powered by socket.io (https://socket.io/), adding reliability and data consistency features in comparison to raw WebSockets, and a custom plugin system providing the interface with laboratory devices. SOPs are implemented using the YAML (http://yaml.org/) format. A custom human-readable syntax was developed that fits the needs of standard laboratory workflows.

A single plugin interfacing with a photometer (ThermoFisher Scientific GENESYS™ 10S UV/VIS) was implemented, which provides functions for setting the wavelength, choosing a turret position, blanking the spectrophotometer and measuring an absorbance value. The photometer was integrated using its integrated USB-RS232 interface. It was directly connected to the backend server via a USB-A/USB-B cable. Port commands were provided by the manufacturer (ThermoFisher Scientific Corp., USA).

The system runs on a Dell OptiPlex 3050 Micro (Dell Technologies Inc., USA) with an Intel Core i3-7100T, 16 GB of DDR4 RAM and a 500 GB SSD. Linux Ubuntu 16.04 is installed as an operating system.

2.2 Front-end applications

Two clients have been developed for communicating with the backend application: A command-line tool for inserting and managing SOPs and an Android application for displaying and executing SOPs.

The command-line tool is a JavaScript application running on NodeJS (https://nodejs.org/en/). It uses the axios (https://github.com/axios/axios) library for HTTP communication and supports inserting, deleting and listing protocols that are available via the backend. Protocols are specified in a simplified YAML syntax that is

mapped to the expected JSON syntax before inserting them via the provided backend HTTP/JSON API. It also automatically uploads associated images to the backend before inserting the protocol.

The Android application was implemented using Android Studio (Alphabet Inc., USA) and the Kotlin programming language (https://kotlinlang.org/).

Internally the Android application uses a reactive-programming approach powered by RxJava (https://github.com/ReactiveX/RxJava) and uses square's moshi (https://github.com/square/moshi) and retrofit (https://github.com/square/retrofit) for JSON decoding and encoding. It also uses square's Picasso (https://github.com/square/picasso) library for downloading and displaying images and socket.io-java-client (https://github.com/socketio/socket.io-client-java) library for interfacing with the WebSocket API provided by the backend. MxParser (http://mathparser.org/) and MathView (https://github.com/Nishant-Pathak/MathView) are used for calculating and rendering mathematical equations. All SOP slides for the individual working steps were prepared using Microsoft Visio 2016 and saved as .png-files with a resolution of 640 × 360 pixels. Different SOPs for the analysis of beer were visualized and corresponding SOP YAML files were created.

The functionality of the Android client application was evaluated using smartglasses Vuzix M300 (Vuzix Corp., USA), which uses specific voice commands to trigger the progress of the SOPs, and a Samsung Galaxy Tab S2 T719N (Samsung Corporate Group, South Korea), which uses graphical touch buttons to move forward and backward through the SOPs.

2.3 Beer analysis assays

Four different MEBAK (Central European Commission for Brewing Analysis) analyses have been transferred into interactive smartglasses-based illustrated SOPs. They have been tested for function and can be carried out using the proposed hard- and software solutions that are freely available. The methods include the spectrophotometric determination of beer color (MEBAK method 2.12.2), the spectrophotometric determination of bittering units in beer (MEBAK method 2.17.1), the spectrophotometric determination of total carbohydrates in beer (MEBAK method 2.6.4.1.1) and the spectrophotometric determination of free nitrogen in beer (MEBAK method 2.6.4.1.1) [14]. Corresponding YAML SOPs are made freely available on GitHub [15]. For detailed procedures and used material see supplemental material.

2.4 System benchmark and user study

To benchmark and give a first insight about the performance of the developed system, 12 graduate students in the field of biotechnology and chemistry (age 23 to 29) were divided into two groups. The first group of six performed MEBAK method 2.12.2 (spectrophotometric determination of beer color) using a conventional paper-based SOP (see supplemental material for details) and the second group of six performed the same experiment with the help of a smartglasses-based SOP. Both groups performed the experiment using the same laboratory working place and instruments (see supplemental material). Experiments were performed in a conventional laboratory surrounding with active exhaust ventilation (50–55 dB; equal to the sound level of a normal conversation). Test persons that executed the paper-based SOP received a calculator and a pen. Neither of the participants had previous knowledge about the performed experiment or the workflow. All participants performed the experiment for the first time to exclude learning effects. The laboratory training status of the test persons was expected to be similar, due to a similar practical education. Prior to the experiment, the group that executed the smartglasses-based SOP received a one-minute introduction on which voice commands to use to operate the application. In addition, the test persons were given time to adjust the smartglasses to their eyes. The experiment execution times were measured using a conventional smartphone timer. The timer was started after turning over the paper-based SOP or using the "select" command within the smartglasses SOP application respectively. The timer has been stopped after the test persons finished the paper-based SOP by saying "finished", or used the command "end protocol" within the smartglasses-based SOP respectively.

A user study was performed with the previously mentioned 12 graduate students. All users performed MEBAK method 2.12.2 (spectrophotometric determination of beer color) using the voice-controlled smartglasses application. Afterwards, they were asked to fill out a survey that included questions about themselves, their opinion about the usability of the utilized application and their impression of the technology (see supplemental material for the questionnaire and detailed results).

3 Results and discussion

3.1 Established server environment

The established server environment that handles all inquiries from clients provides both a JSON/HTTP API (Application Programming Interface) for retrieving and creating SOPs and a separate WebSocket API that manages all instrument measurements that are triggered via the backend (see Fig. 2).

This separation is due to the duality of workflows; they contain static data (e.g. the instructions for each step of the SOP or an equation), that can be fetched beforehand, as well as dynamic data (e.g. measurement results) that can only be made accessible after a certain progression in the SOP was reached.

Instrument integration into the established setup was done via the native USB-RS-232 interface of the spectrophotometer and a wired USB connection to the backend server. A plugin structure was worked out to enable flexible device integration. This means a new plugin needs to be created for every new device that should be integrated. Plugins contain information about the device interface, the communication protocol and specific command strings that are mapped to a keyword within the SOP syntax to trigger the command (see chapter 3.3). This approach allows the use of a wide range of common communication standards including MQTT, Modbus, and TCP/IP (see GitHub repository for the used RS-232 JavaScript plugin structure) [15–18]. Wireless communication standards like Bluetooth of WiFi can also be implemented (see

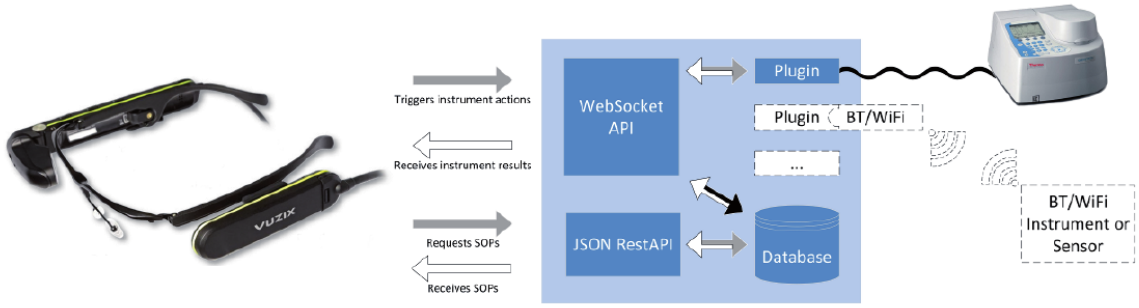


Fig. 2 **Architecture and data flow of the established setup. Gray arrows indicate data requests and white arrows indicate the return of the requested data. The black arrow specifies the storage of instrument and process data in the database. White boxes with dashed lines state flexible plugins that can be implemented into the architecture**

Fig. 2) if appropriate hardware standards are met on both the back end and instrument site of the setup.

With every start of the developed backend application, the WebSocket stack loads all plugin JavaScript files in a predetermined directory and registers their name and provided function with a custom plugin manager. When a client wants to trigger a measurement it sends a request containing a valid action ID in a custom WebSocket channel. The backend then checks if this request is valid and synchronously returns an associated unique result ID to the client, therefore allowing the client to issue multiple requests in parallel. It then asynchronously performs the measurement by calling the corresponding function on the loaded plugin and creates a database entry (IOResult) for the result containing the same unique ID that was returned to the client. The client receives the result value and ID asynchronously via a WebSocket channel and then updates its interface. We chose an asynchronous approach instead of a polling approach to improve application and backend performance and scalability. Although the workflow data is linked to an automatically generated ID that is traceable within the relational database, the registration of a custom ID might be desirable. Depending on the technology that is used to trace samples within the laboratory or company, different solutions are implementable. Either the previous scanning of sample QR or barcodes or the recording of voice and subsequent speech-to-text processing are options to store individual sample IDs [19, 20]. Additionally, the IDs in the database can be modified using standard SQL (Structured Query Language) operations. The provided SQL API allows for the integration of the database into third party software like LIMS and ERP (entertprise resource planning) systems that enable the

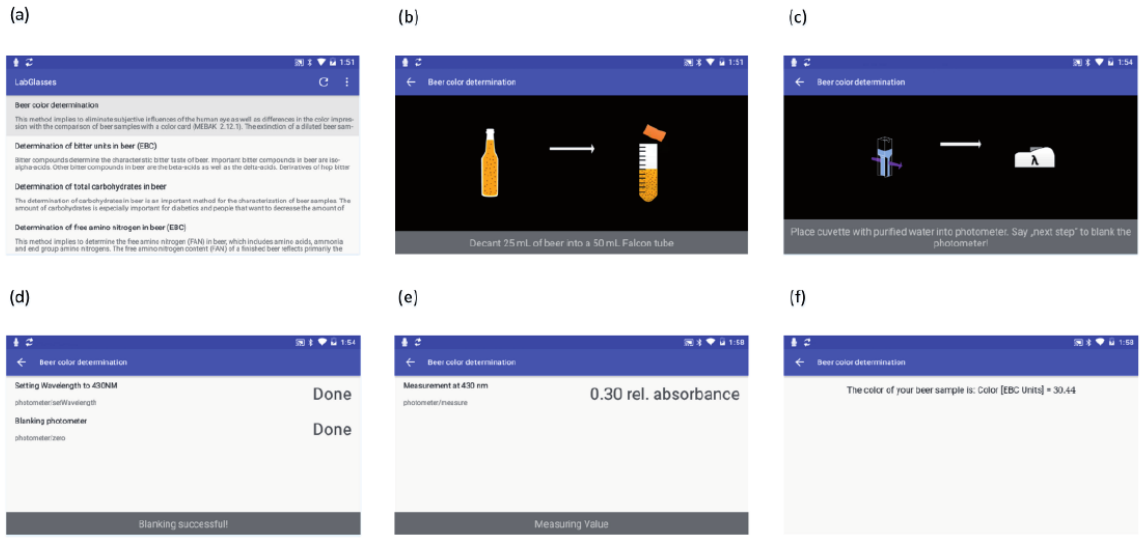


Fig. 3 **Exemplary screens showing the progress of the SOP for beer color determination displayed into the experimenter's field of view. (a) Master screen that is shown after opening the app and that lists the currently available SOPs and a short description (all implemented SOPs are shown); (b) Ordinary instruction that is not mapped to an action (instruction text is shown in the gray box under the instruction pictograms); (c) Ordinary instruction that precedes an instruction that triggers an action; (d) An instruction that triggers an action and shows the successful execution (setting of a wavelength and blanking); (e) Display of a successful measurement and the corresponding relative absorbance units; (f) Display of the experimental results using an equation that is defined within the SOP**

Table 1 Implemented voice commands that allow the navigation within the developed client Android application

Voice command	Triggered action
"Up"	Navigates up in the master detail screen
"Down"	Navigates down in the master detail screen
"Select"	Selects and starts an SOP within the master detail screen
"Next Step"	Triggers the next step of the SOP
"Previous Step"	Goes back to the previous step of the SOP
"Next Result"	Navigates forward in the result menu in multipath SOPs
"Previous Result"	Navigates backward in the result menu in multi-path SOPs
"End protocol"	Finishes the protocol after the last experimental instruction and goes back to the master detail screen
"Go home"	Ends the application

creation of experimental reports and the monitoring of experimental progresses [21].

3.2. Front-end application for smartglasses

The developed application is running on Vuzix M300 smartglasses (Vuzix Corp., USA) as well as all other Android devices above API Level 23 (restricted due to Vuzix development tools). It features a Master-Detail application flow (see Fig. 3, screen a) allowing the user to first select a protocol and then guides the user through the instructions, even allowing voice navigation on the Vuzix M300 smartglasses (see Table 1 for implemented voice commands). Instructions are displayed according to their type, i.e. timer instructions display a specified countdown, equation instructions display a mathematical equation that can contain previous measurement results and specific calculated results, and normal instructions are rendered with an associated image plus a text overlay containing the instruction text (see Fig. 3, screen 2). Additionally, if measurements (actions) are associated with the instruction, a list of pending measurements is displayed, automatically executed and finally displayed to the user (see Fig. 3, screens 4 and 5). The application prevents the progression of the SOP if a step is not finished, such as a countdown has not run out yet or a measurement failed. If a multi-path instruction is detected the user is prompted to select a result that matches his experimental progress. The correct following instructions are then displayed to the user and irrelevant instructions are hidden.

SOP pictograms were custom-made for all experimental steps and integrated as .png-files. A standard graphic format, which allows the use of nearly every image editing program.

The evaluation of the application was done with test users that are used to performing laboratory experiments and have a similar professional background. The main focus was to reach a high comprehensibility of the SOPs and an easy handling of the client app to achieve an independent performance of an experiment. It turned out that the used pictograms are comprehensible and the voice commands could be processed with high accuracy. All test persons could perform the implemented SOPs completely self-reliant (see 3.4).

3.3 SOP creation and syntax

To ensure the creation of new SOPs by untrained users without any programming skills, a human-readable, easy to understand markup language format was chosen. The YAML format (YAML Ain't Markup Language) allows the straightforward construction of SOPs, where a list of arrays forms a specific SOP step.

Each SOP has a unique ID, by which it can be identified and retrieved by the Android client-application, as well as a name and

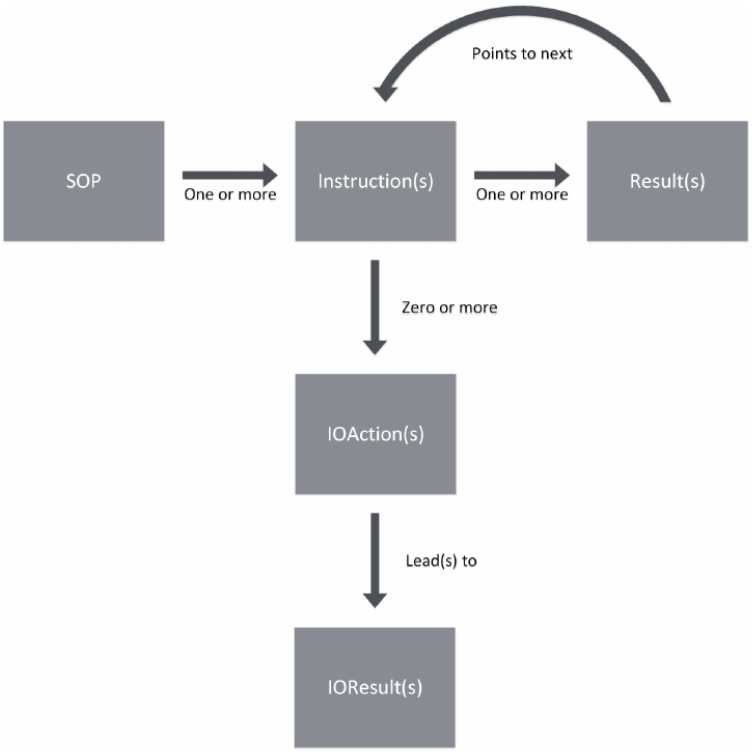


Fig. 4 Flowchart of an SOP. An SOP consists of one or more instructions. An instruction leads to one or more results (depending if it is a multipath experiment or not). An instruction can contain zero or more IOActions. IOActions are instrument actions that are triggered at a defined instruction. These IOActions lead to corresponding IOResults (e.g. a successful measurement or the adjustment of instrument parameters)

```
name: Beer color determination
description: This method implies to eliminate subjective influences of the human eye as well as differences in the color
impression with the comparison of beer samples with a color card (MEBAK  2.12.1). The extinction of a diluted beer sample is
measured in a 10-mm-cuvette at a wavelength of 430 nm. Color giving melanoides and melanoidines that are formed in the
brewing process absorb at 430 nm. The color in EBC units can be calculated by using an appropriate factor.
imageBasePath: beercolor/ # a path relative to the global image base path
instructions: # we will refer to instruction as steps and vice versa for the length of the example
    - Step 1 : # this is the instruction identifier which is later used for target this instruction with a result
        description: Beer color determination.\n Say "next step" to advance to the next work instruction and „previous
       step" to go to the previously shown work instruction!
        results:
            - add a description for a result here: Step 2
    - Step 2 :
        description: Put on Gloves!
        results:
            - add a description for a result here: Step 3
    - Step 3 :
        description: Decant 25 mL of beer into a 50 mL Falcon tube
        results:
            - add a description for a result here: Step 4
        ...

    - Step 10 :
        description: Place cuvette with purified water into photometer. Say „next step" to blank the photometer!
        results:
            - add a description for a result here: Step 11
    - Step 11 :
        description: Blanking successful!
        actions: # actions describe anything that happens with an external device
            - plugin: photometer
              action: setWavelength
              wavelength: 430
              humanReadableName: Setting Wavelength to 430NM
              unit: null
            - plugin: photometer # the name of the corresponding plugin in labor-api (required)
              action: zero # each plugin supports certain actions (required)
              humanReadableName: Blanking photometer
              unit: null
        results:
            - add a description for a result here: Step 12
    - Step 12 :
        description: Remove previously inserted cuvette and place cuvette with beer sample into photometer
        results:
            - add a description for a result here: Step 13
    - Step 13 :
        description: Say „next step" to measure!
        results:
            - add a description for a result here: Step 14
    - Step 14 :
        description: Measuring Value
        actions:
            - plugin: photometer
              action: measure
              humanReadableName: Measurement at 430 nm
              equationIdentifier: a
              unit: rel. absorbance
        results:
            - add a description for a result here: Step 15
    - Step 15 :
        description: The color of your beer sample is: \n\nColor [EBC Units]:
        equation: a*25*4
        results:
            - add a description for a result here: Step 16
    - Step 16 :
        description: Dispose all samples. \n The experiment has finished!\n Say \"end protocol\" to exit the SOP!
        results:
            - experiment is finished: null # if you specify null as the next instruction, the labor-api-cli will know that
              this is your last instruction.
```

Fig. 5 **Structure of an SOP using the suggested YAML structure. The SOP describes the spectrophotometric determination of beer color (MEBAK 2.12.2, see chapter 2.3.1). The three dots after step 3 represent steps 4–9, which are ordinary instructions any instrument actions involved. Gray highlighted text shows the name of the SOP and a description of the process that is shown in the master screen (see Fig. 3 a), and the source directory of the image data. Italic and bold text marks descriptions that are shown in the smartglasses' display as instruction text under the individual step image. They can also contain equations (see step 15) that process previously recorded data (so-called IOResults). Underlined text represents instrument actions (so-called IOActions), which portray device events (e.g. measurements or adjustments). They can be linked to an equation identifier, which enables the processing of instrument data that is recorded during the action in a later step (see step 15)**

a description. This unique ID is generated when the protocol is initially uploaded with the developed command-line tool. In addition, each SOP contains an array of instruction structures, where each instruction represents a step in the source SOP (see Fig. 4). An instruction contains a unique ID, a description, an optional image, one or more results, zero or more IOActions and optionally an equation or a timer sequence (for incubation or waiting sequences) (see Fig. 5). Each result contains a description, an optional image and a target instruction identified by its unique ID. Instructions and results from a circular progression structure allowing it to effortlessly represent multi-path SOPs, where experiment progression depends on intermediary experimental results (e.g. specific color or turbidity changes that require different experimental paths) (see Fig. 4).

IOActions contain a unique ID, a plugin name, a plugin function name, optional plugin arguments, a human-readable name, an optional unit, and an optional equation identifier. They represent external laboratory instrument actions associated with this instruction. Plugin name and plugin action names map to plugins provided by the custom plugin system through the WebSocket API (see Fig. 2). The equation identifier can be used to reuse the IOResult, which was retrieved in the associated instruction in later instructions that might have an equation associated with them (e.g. a calculation at the end of the experiment). Specifying an equation on an instruction changes the type of the instruction to display a calculated mathematical expression instead of a description and image. Similarly, specifying timer duration on an instruction changes the type of the instruction so that the client will display a countdown. Pictograms are associated with the instruction step by naming the specific pictograms after the step (e.g. the pictogram for step 2 would be implemented as 2.png). If no image data is available, only the text of the step description is shown in the Android client app.

The four above mentioned MEBAK analyses (see chapter 2.3.) have been implemented into the setup. Detailed illustrated SOPs were worked out and have been tested (see online repository for details) [15].

Although the first step to a user-friendly creation of SOPs for the developed laboratory environment is done, future efforts include the establishment of an “SOP wizard”. This tool shall make the creation of SOPs even easier, by providing the user a graphical user interface (GUI) to enter the descriptions, actions, and images.

3.4 System benchmark and user study

A benchmark of the system was performed by letting two groups of test users perform the SOP for the spectrophotometric determination of beer color (see supplemental material for details). One group used a paper-based SOP (see supplemental material) and the other group performed the experiment using the smartglasses-based SOP [15]. A total of 12 test users (6 female and 6 male users) were used for the first benchmark of the system (see 2.4 for more details). Users that are not familiar with the analysis method were chosen as test persons, a method previously reported to benchmark assisted industrial assembly workflows [22]. A benchmarking study including method experienced laboratory technicians might have led to a different outcome, though, and is intended for future studies.

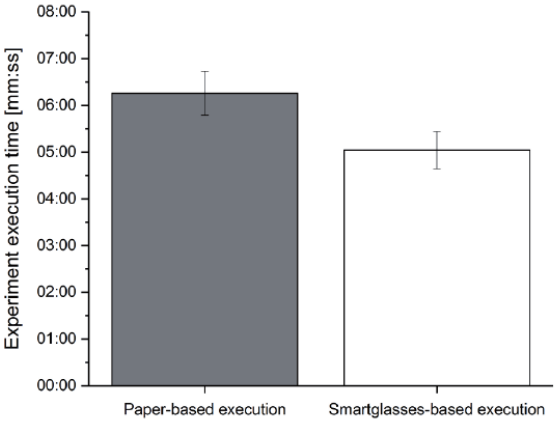


Fig. 6 **Average experiment execution times of the paper-based SOP execution of the spectrophotometric determination of beer color (MEBAK method 2.12.2) (gray column) and the smartglasses-based SOP execution of the same experiment (white column)**

The average execution time of the paper-based SOP was determined as 6:16 min ± 28 s (median: 6:14 min), whereas the smartglasses-based execution time of the experiment could be stated as 5:03 min ± 24 s (median: 5:01 min; see Fig. 6).

This results in an average process efficiency boost of 19.4 % by using the presented infrastructure. Due to the automated spectrophotometer actions and the automated calculation of the experimental result, time could be saved. It has to be stated that the performed experiment only involves three instrument actions and only one measurement and calculation. More complex smartglasses-based SOPs could even increase the workflow efficiency compared to a paper-based SOP execution. Because all test persons performed the experiment for the first time, the developed system might as well be an interesting tool to reduce initial training phases for experimental workflows.

After performing the paper-based SOP, the first group of test users was given the chance to perform the previously made experiment using the smartglasses. Afterwards, all users participated in a survey to rate the overall usability of the system and to state their opinion about smartglasses technology in the laboratory (see supplemental material for details). Two of the study participants had previously worn smartglasses, for demonstration purposes only, though. The overall usability of the system received an average rating of 8.75 ± 0.75 on a scale from 1 (bad) to 10 (very good; median: 9), which states a user-friendly and intuitive application design. Furthermore, 4 out of 12 users stated that they have privacy concerns about using smartglasses as a support for their laboratory work. Yet, all participants declared that they think smartglasses have the potential to increase the efficiency of laboratory work. 25 % of the participants think that it takes a decade until the presented technology is widely available in laboratory environments, whereas 50 % think the technology will be already present in 3 to 5 years. All but one participant of the survey reported that they sometimes use their personal smartphone as a support during laboratory work (e.g. to look things up, or to calculate dilutions, etc.). This indicates

that smart devices are already part of the daily laboratory routine. New solutions like smartglasses combine smartphone features with a hands-free operating mode. Once this technology is more widespread it has not only the chance to change the work efficiency in laboratories but also other professional environments [23].

5 Conclusion

We established a flexible digital laboratory infrastructure that enables the bidirectional communication between smart devices and laboratory instruments. Additionally, we were able to show the applicability of smartglasses as a voice-controlled supporting tool for analysis workflows in the brewing laboratory. We proposed an easy to understand and human-readable SOP syntax for an uncomplicated integration of SOPs into the developed infrastructure. Our benchmarks revealed that the established solution could save time compared to a conventional paper-based experiment execution and documentation. Furthermore, we could show that the developed smartglasses application is user-friendly and features an easy operation.

The described architecture has the potential to significantly ease the work burden of performing and documenting standard experimental processes since all device parameters and measurement results are automatically saved within a designated database. The integration of the system into commercially available electronic laboratory notebooks or laboratory information and management sytsems can be realized using the integrated SQL database API, which enables the automated generation of a process report. Also the communication with common ERP systems can be performed to monitor the progression of workflows. This could lead the way to data documentation being in accordance with good manufacturing and good laboratory practice standards. Additionally, the proposed solution might be of interest for education and training purposes since an easy to handle application for the step-by-step experiment performance is provided.

Smartglasses just recently emerged in science and other professional fields. To unleash their full potential within the laboratory and other environments, a holistic digitization approach must be pursued. The IT infrastructure, as well as the instrument infrastructure, needs to be set up or updated. To simplify integration work, instrument manufacturers need to move together to agree upon consistent instrument communication standards. This results in less documentation work, fewer documentation errors and inevitable efficiency boosts. Although the performed process and hardware evaluation can overall be described as positive, future work focuses on further reliability tests, the scale-up of the instrument setup and the determination of the technology acceptance of test persons from a more diverse field of professional backgrounds [24]. Another important step that needs to be performed to use the proposed system within the industry is the validation of the architecture and all of its components [25, 26]. Furthermore, controversial issues regarding the technology, including the potential violation of personal rights due to the integrated camera system and the possible negative health effects resulting from long-term usage of wireless technologies, need to be discussed at length [27–29]. We decided to upload the project code to GitHub, to enable other researchers to profit from our findings and to evaluate the architecture in their laboratory surroundings and to customize the software to fit their needs[15].

Funding

This work was financially supported by the Bavarian Ministry of Economic Affairs and Media, Energy and Technology within the Information and Communications Technology program (Grant number: IUK-1504-0006//IUK470/001).

Acknowledgments

The authors thank Ethan Overfelt for proofreading the manuscript and all test users for their participation.

Conflicts of Interest

The authors declare no conflict of interest. The funders had no role in the design of the study; in the collection, analyses, or interpretation of data; in the writing of the manuscript, and in the decision to publish the results.

6 References

1. Sutherland, I.E.: A head-mounted three dimensional display, Proceedings of the December 9-11, 1968, fall joint computer conference, part I on - AFIPS '68 (Fall, part I), ACM Press, New York, New York, USA, 1968, p. 757.
2. Furht, B.: Handbook of Augmented Reality, Springer New York, New York, NY, 2011.
3. Mann, S.: Wearable computing: A first step toward personal imaging, Computer, **30** (1997), no. 2, pp. 25-32.
4. Muensterer, O.J.; Lacher, M.; Zoeller, C.; Bronstein, M. and Kübler, J.: Google Glass in pediatric surgery: An exploratory study, International Journal of Surgery, **12** (2014), no. 4, pp. 281-289.
5. Liebert, C.A.; Zayed, M.A.; Aalami, O.; Tran, J. and Lau, J.N.: Novel Use of Google Glass for Procedural Wireless Vital Sign Monitoring, Surgical Innovation, **23** (2016), no. 4, pp. 366-373.
6. Mitrasinovic, S.; Camacho, E.; Trivedi, N.; Logan, J.; Campbell, C.; Zilinyi, R.; Lieber, B.; Bruce, E.; Taylor, B.; Martineau, D.; Dumont, E.L.P.; Appelboom, G. and Connolly, E.S.: Clinical and surgical applications of smart glasses, Technology and Health Care, **23** (2015), no. 4, pp. 381-401.
7. Farrell, W.A.: Learning Becomes Doing: Applying Augmented and Virtual Reality To Improve Performance, International Society for Performance Improvement, **57** (2018), no. 4, pp. 19-28.
8. Young, H.A.: Scientific Apps are here (and more will be coming), Cytokine, **59** (2012), no. 1, pp. 1-2.
9. Austerjost, J.; Porr, M.; Riedel, N.; Geier, D.; Becker, T.; Scheper, T.; Marquard, D.; Lindner, P. and Beutel, S.: Introducing a Virtual Assistant to the Lab: A Voice User Interface for the Intuitive Control of Laboratory Instruments, SLAS Technology, **23** (2018), no. 5, pp. 476-482.
10. Cortazar, B.; Koydemir, H.C.; Tseng, D.; Feng, S. and Ozcan, A.: Quantification of plant chlorophyll content using Google Glass, Lab Chip, **15** (2015), no. 7, pp. 1708-1716.
11. Feng, S.; Caire, R.; Cortazar, B.; Turan, M.; Wong, A. and Ozcan, A.: Immunochromatographic diagnostic test analysis using google glass, ACS Nano, **8** (2014), no. 3, pp. 3069-3079.

12. Austerjost, J.; Marquard, D.; Raddatz, L.; Geier, D.; Becker, T.; Scheper, T.; Lindner, P. and Beutel, S.: A smart device application for the automated determination of E. coli colonies on agar plates, Engineering in Life Sciences, **17** (2017), no. 8.
13. Strzys, M.P.; Kapp, S.; Thees, M.; Klein, P.; Lukowicz, P.; Knierim, P.; Schmidt, A. and Kuhn, J.: Physics holo.lab learning experience: Using Smartglasses for Augmented Reality labwork to foster the concepts of heat conduction, European Journal of Physics, **39** (2017), no. 3, pp. 1-12.
14. Jacob, F.: Wort, beer, beer-based beverages : collection of brewing analysis methods of the Mitteleuropäische Brautechnische Analysenkommission, Selbstverl. der MEBAK, 2013.
15. Austerjost, J. and Bargholz, M.: LabGlasses repository on GitHub, https://github.com/tciluh?tab=repositories, accessed 24 July 2018.
16. Hunkeler, U.; Truong, H.L. and Stanford-Clark, A.: MQTT-S – A publish/subscribe protocol for Wireless Sensor Networks, 2008 3rd International Conference on Communication Systems Software and Middleware and Workshops (COMSWARE '08), (2008), pp. 791-798.
17. Fovino, I.N.; Carcano, A.; Masera, M. and Trom-Betta, A.: Design and implementation of a secure Modbus protocol, IFIP Advances in Information and Communication Technology, Springer, Berlin, Heidelberg, 2009, pp. 83-96.
18. Forouzan, B.A.; A./Fegan, B. and Chung, S.: TCP/IP protocol suite, McGraw-Hill, 2003.
19. Tarjan, L.; Šenk, I.; Tegeltija, S.; Stankovski, S. and Ostojic, G.: A readability analysis for QR code application in a traceability system, Computers and Electronics in Agriculture, **109** (2014), pp. 1-11.
20. Schalkwyk, J.; Beeferman, D.; Beaufays, F.; Byrne, B.; Chelba, C.; Cohen, M.; Kamvar, M. and Strope, B.: "Your Word is my Command": Google Search by Voice: A Case Study, Advances in Speech Recognition, Springer US, Boston, MA, 2010, pp. 61-90.
21. Saf'yanov, A.S.; Tereshchenko, A.G.; Tereshchenko, V.A.; Yanin, A.M.; Yunak, A.L. and Tereshchenko, O. V.: Information interaction between LIMS and corporate information system elements at enterprise, Automation and Remote Control, **73** (2012), no. 4, pp. 760-764.
22. Funk, M.; Kosch, T.; Greenwald, S.W. and Schmidt, A.: A benchmark for interactive augmented reality instructions for assembly tasks, Proceedings of the 14th International Conference on Mobile and Ubiquitous Multimedia – MUM '15, (2015), pp. 253-257.
23. Kim, S.; Nussbaum, M.A. and Gabbard, J.L.: Augmented Reality "Smart Glasses" in the Workplace: Industry Perspectives and Challenges for Worker Safety and Health, IIE Transactions on Occupational Ergonomics and Human Factors, **4** (2016), no. 4, pp. 253-258.
24. Rauschnabel, P.A. and Ro, Y.K.: Augmented reality smart glasses: an investigation of technology acceptance drivers, International Journal of Technology Marketing, **11** (2016), no. 2, p. 123.
25. Hoffmann, A.; Kähny-Simonius, J.; Plattner, M.; Schmidli-Vckovski, V. and Kronseder, C.: Computer system validation: An overview of official requirements and standards, Pharmaceutica Acta Helvetiae, **72** (1998), no. 6, pp. 317-325.
26. López, O.: 21 CFR Part 11 – Complete Guide to International Computer Validation Compliance for the Pharmaceutical Industry, CRC Press, 2004.
27. Mann, S.: Eye Am a Camera : Surveillance and Sousveillane, Time, http://techland.time.com/2012/11/02/eye-am-a-camera-surveillance-and-sousveillance-in-the-glassage/, accessed 23 July 2018.
28. World Health Organization: Electromagnetic fields and public health: mobile phones, http://www.who.int/en/news-room/fact-sheets/detail/electromagnetic-fields-and-public-health-mobile-phones, accessed 23 July 2018.
29. Baan, R.; Grosse, Y.; Lauby-Secretan, B.; El Ghissassi, F.; Bouvard, V.; Benbrahim-Tallaa, L.; Guha, N.; Islami, F.; Galichet, L. and Straif, K.: Carcinogenicity of radiofrequency electromagnetic fields, The Lancet Oncology, **12** (2011), no. 7, pp. 624-626.

Received 25 October 2018, accepted 2 February 2019

3.6 IMPLEMENTIERUNG EINER DIGITALEN LABORINFRASTRUKTUR

Die oben beschriebenen Arbeiten und Veröffentlichungen behandelten Teilaspekte des digitalisierten Labors. Sie legen den Fokus auf verschiedene Technologien und beschreiben die notwendigen Entwicklungen und Vorgänge zur Implementierung im Detail. Im folgenden Forschungsartikel werden diese Bausteine zum Gesamtbild eines digitalisierten biotechnologischen Labors zusammengefügt. Die Veröffentlichung beschreibt die unterschiedlichen Technologien im Kontext und legt besonderen Wert auf die Wechselwirkungen zwischen den Teillösungen.

Neben der standardisierten Gerätanbindung und Beschreibung des zentralen Laborservers wird mit Hilfe eines praxisnahen Beispiels die Umsetzung eines Labordigitalisierungsprojektes gezeigt und die einzelnen notwendigen Schritte erläutert. Es wird ein einfacher Beispiel-Arbeitsablauf vorgestellt und das Vorgehen zur Umsetzung dieses Ablaufes im digitalisierten Labor skizziert.

Da neben der Geräteansteuerung und Datenhaltung dem Prozessleitsystem und der Nutzerinteraktion im digitalisierten Labor eine besondere Rolle zukommt, werden die zu diesem Zweck entwickelten Systeme in den unterstützenden Informationen detailliert beschrieben und die Anwendung mit einfachen Beispielen praxisnah erklärt (Siehe Kapitel 3.6.1, Seite 108). Abbildung 26 zeigt beispielhaft ein Ablaufskript für das digitalisierte Labor. Durch die leicht verständliche, skript-artige Beschreibung der Prozesse können auch Programmieranfänger und Studenten die Bedienung des digitalisierten Labors schnell erlernen. Interaktion mit den Laborgeräten und dem Laborbenutzer wird auf einfache und unkomplizierte Art formuliert. Dabei zeichnen sich die zu verwendenden Befehle durch einen hohen Abstraktionsgrad aus, so dass die technischen Prinzipien der Geräteinteraktion und des Datentransfers dem Methoden-Ersteller nicht im Detail bekannt sein müssen.

Durch diese Vorgehensweise könnte in einem weiteren Schritt auch eine grafische Oberfläche zum Prozessdesign leicht integriert werden.

```
1  module usecases.example;
2  import controlflow.usecase;
3  mixin RegisterAllUseCases;
4
5  @UseCase void example()
6  {
7      flow.initStorage([UIDev.glass], "TCI", "example", "layouts/example");
8
9      flow.step("prepare", {
10         sendLayout("prepareCentrifugeUI");
11         uiEvents("next").wait();
12     });
13
14     flow.step("centrifuge", {
15         auto cmd = command("centrifuge", "Start", 2000, 60, 20);
16         cmd.wait();
17     });
18
19     flow.step("timer", {
20         sendLayout("timerUI");
21         wait(timer("wait", 25.seconds));
22     });
23 }
```

ABBILDUNG 26: BEISPIEL PROZESS-MANAGEMENT-SKRIPT ZUR VERDEUTLICHUNG DER LEISTUNGSFÄHIGKEIT DES SYSTEMS. GERÄTESTEUERUNG UND NUTZERINTERAKTION KANN IN EINER SKRIPT-ARTIGEN, LEICHT ERLERNBAREN FORM PROGRAMMIERT WERDEN.

Received: 28 July 2020 | Revised: 25 November 2020 | Accepted: 25 November 2020

DOI: 10.1002/elsc.202000053

Engineering in Life Sciences

RESEARCH ARTICLE

Implementing a digital infrastructure for the lab using a central laboratory server and the SiLA2 communication standard

Marc Porr | **Ferdinand Lange** | **Daniel Marquard** | **Laura Niemeyer** | **Patrick Lindner** | **Thomas Scheper** | **Sascha Beutel**

Institute of Technical Chemistry, Leibniz University Hannover, Hannover, Germany

Correspondence
Dr. Sascha Beutel, Institute of Technical Chemistry, Leibniz University Hannover, Callinstraße 5, 30167 Hannover, Germany.
Email: beutel@iftc.uni-hannover.de

This article is dedicated to Prof. Thomas Bley on the occasion of his 70th birthday.

Funding information
Digitalisierung in der Industriellen Biotechnologie (DigInBio), Grant/Award Number: FKZ 031B0463C

Abstract

In this report, a fully integrated solution for laboratory digitization is presented. The approach presents a flexible and complete integration method for the digitally assisted workflow. The worker in the laboratory performs procedures in direct interaction with the digitized infrastructure that guides through the process and aids while performing tasks. The digital transformation of the laboratory starts with standardized integration of both new and "smart" lab devices, as well as legacy devices through a hardware gateway module. The open source Standardization in Lab Automation 2 standard is used for device communication. A central lab server channels all device communication and keeps a database record of every measurement, task and result generated or used in the lab. It acts as a central entry point for process management. This backbone enables a process control system to guide the worker through the lab process and provide additional assistance, like results of automated calculations or safety information. The description of the infrastructure and architecture is followed by a practical example on how to implement a digitized workflow. This approach is highly useful for – but not limited to – the biotechnological laboratory and has the potential to increase productivity in both industry and research for example by enabling automated documentation.

KEYWORDS
internet of things, laboratory device communication, laboratory digitization, laboratory network, SiLA2

Abbreviations: AnIML, analytical information markup language; API, application programming interface; CIL, common intermediate language; FAIR, findability, accessibility, interoperability, reusability; gRPC, google remote procedure call; (G)UI, (graphical) user interface; ID, (unique) identifier; IPC, inter process communication; JSON, javascript object notation; NFDI, nationale forschungsdateninfrastruktur; PCR, polymerase chain reaction; POSIX, portable operating system interface; Proxmox VE, proxmox virtual environment; QR code, quick response code; RS232, recommended standard 232; REST, representational state transfer; RPC, remote procedure call; SiLA2, standardization in lab automation 2; SSH, secure shell; TCP/IP, transmission control protocol/internet protocol; USB, universal serial bus; (W)LAN, (wireless) local area network; XML, extensible markup language

1 | INTRODUCTION

In this article, a concept for digital integration of a biotechnological laboratory is presented. The main goals are centralized and automatic acquisition of data and metadata, enabling assistant technologies for the researcher and automated documentation. In contrast to decentralized approaches [1, 2], this method is focused on a centralized control- and data-management system that interacts with human inputs and orchestrates procedures based on the results and outcomes of previous steps. Most digitization concepts available focus on highly automated laboratories [3–5] with processes that require only a small amount of human-machine interaction [6]. This creates automated and smart "digitized islands" in the laboratory that are surrounded by an ocean of dumb lab equipment and devices. Often these islands are not standing separate but other devices and manual procedures are needed for additional steps. Even though this user interaction problem was named by Frey in 2004 [7], interaction of the digitized lab with humans is still a daunting task today. Especially in the biotechnological lab, where complex protocols are carried out and a lot of additional information has to be processed by the researcher, a method for adding digital support seems promising.

In this approach, all devices and resources can be accessed in a central place. The laboratory server guaranties a full record of measurements, results and events. This enables possibilities for certification – for example GxP (Good Manufacturing/Laboratory Practice) compliance [3]. Also implementing FAIR (Findable, Accessible, Interoperable, Reusable) data principles [8], which are for example required by the NFDI (Nationale Forschungsdateninfrastruktur), is possible.

For good maintainability, a micro-service based architecture was chosen. The approach focuses on standardized communication of all components. This enables good horizontal scaling, which is important in digital transformation of existing infrastructures [9]. One central Representational State Transfer (REST) Application Programming Interface (API) is used for operation procedure orchestration and data requests. All devices are connected using the Standard in Lab Automation 2 (SiLA2) standard for laboratory device communication[1]. The usage of one single standard highly increases the flexibility and usability of the setup [10]. The first stable version of the SiLA2 standard was released in 2019 and several implementations in different programming languages are available today[2]. The "race for the lab communication standard" is on [11] and nobody can predict which standard will be widely adopted in the future. However, with its open source concept and good documentation, SiLA2 offers some significant advantages for the developer that aims on integrating existing devices without a common communication principle. For these kind of devices, a "translator" for SiLA2 must be used [5]. This, at first sight, might look like only shifting implementation efforts. However, as no common standard for lab device integration exists, the focus on one standard wants to empathize the need for standardization and flawless integration. Furthermore, from a software architect's point of view, the separation of concerns is an important pattern in complex systems. When using a standard, the laboratory server does not need to know about all the different protocols for all devices. It can be kept simple in both design and implementation, which is important for good scalability and maintainability. This approach divides the big integration problem into small separate tasks. All translators share a common structure and form a swarm

PRACTICAL APPLICATION

The presented architecture can be used to transform a traditional laboratory into a digitized one. It focuses on the interaction with the laboratory worker in processes that cannot – or should not – be automated. Procedures are defined in advance and are carried out in a structured way that ensures a correct workflow. The architecture centralizes information and laboratory device communication in a server, which also hosts a database that stores all data generated in the laboratory. The focus lies on small research or analysis laboratories where procedures are not highly automated, but where data quality and productivity can significantly benefit from digital assistance. This opens up possibilities of interactive workflow guidance for the lab worker, automated result documentation and an improved control over the lab processes. Workers can spend more time on productive tasks when calculations, data analysis and documentation are carried out fully automated in the background.

[1] The open source SiLA2 standard definition documents can be found online. Part A: https://drive.google.com/file/d/1QWrSD4-YBMwT9HTBzJe3TlBbJSGqq_LC/view Part B: https://drive.google.com/file/d/1a9XJnQUHysW6DQGz4j2m1zngLX-pakxo/view Part C: https://drive.google.com/file/d/1dqQTqRN6vyJy6KBISCsnV_qte_kNh62l/view

[2] Both the official reference implementations in several programming languages and also the sila_tecan C# implementation used in this article can be found at: https://gitlab.com/SiLA2

of micro-services that are managed from a central web-interface of the laboratory server. In addition, deployment of these services is automated by a central publish-system. Both the management and publish system, as well as a guide on how to write SILA2 translators as micro-services, were published by the authors before [12] and are available under an open-source license.

Devices that cannot be connected directly to the laboratory network, are integrated using a gateway module. This was described in detail by the authors before [12] and all necessary information and software are available under open licenses.

The flexible nature of the system and the central REST interface of the laboratory server enable easy connecting of different process management tools. Using an appropriate middleware, an electronic lab notebook or any other form of data structuring or data analysis solution can easily be connected to the lab server as well.

The communication of the laboratory system with the researcher is modeled using a standard gateway to all possible user interface devices (UI-devices). For example, these can be head-mounted displays ("smart-glasses"), tablets, etc. Due to its complexity, the user-interface-system and the process management tool co-implemented with this system cannot be described in detail in this article. However, the interested reader can find the details in the supplementary data.

Traditionally, a researcher needs to follow a printed standard operating procedure and must document all steps by hand. Additionally, interpreting, calculating and archiving of result data needs to be carried out manually afterward. Using the presented architecture, these tasks can be automated and standardized. This is possible due to the central data storage and fully flexible availability of data.

This increases data quality [5, 6, 11] as human errors or mistakes in copying notes are much less likely. The automated data management takes the burden of result documentation from the researcher and leaves more time for performing qualified tasks. This may help to overcome the lack of skilled labor currently experienced in the fields of laboratories and research [3, 4, 6, 11] – especially in the biotechnological sector.

2 | MATERIALS AND METHODS

This approach relies on a dependable infrastructure to ensure fast and secure communication of the distributed systems. Wherever possible existing technologies were used. Today's technology is quite capable of performing all relevant tasks in a digitized lab. The current hype in home automation and the "internet of things" shows what can be achieved when manufacturers agree on communication standards [13]. However, such industry standards are not yet present in the laboratory world. Therefore, some effort has to be made to enable devices from different manufacturers to exchange data and commands [5].

2.1 | Hardware integration

Figure 1 shows an overview of the hardware used for the digital network. For communication of digital components Transmission Control Protocol/Internet Protocol based protocols are widely used and easily scalable. Thus, an Ethernet-based local area network with DHCP (Dynamic Host Configuration Protocol) forms an adequate base for lab digitization. Wherever possible wire-based connections were used due to their reliability. User interface devices, like wearables or handheld devices are connected via WLAN (Wireless Local Area Network) access points. User computers, which run protocols and interact with the lab server, are connected to Ethernet ports in the lab or via WLAN.

Lab devices with an Ethernet port can easily connect to the lab network, as no further hardware is needed. Older or simpler devices do not have Ethernet or WLAN capabilities. These devices can usually be connected to a controller or computer by a serial- or Universal Serial Bus (USB)-connection. These connection methods are not easily scalable, not sharable and cable lengths are limited. This means a lab server would have to be connected to every of those devices directly. As the server will usually not be in the same physical location as the devices but in a dedicated server room, connecting legacy devices is a great challenge. A gateway-module was used to integrate such devices into the digitized lab network. The building and programming of this gateway module was described in detail by the authors before [12] and all plans and software are available under an open-source or open-hardware license. The module is part of the laboratory network and connects to the lab device via USB or serial/Recommended Standard 232. Communication from the module to the device is done in whatever protocol is required by the specific device. One module can connect multiple lab devices, which can be desirable if they are placed close together.

2.2 | Data transfer and protocols for device integration

For data transfer, standardized network protocols were used. The lab server offers all functionality via

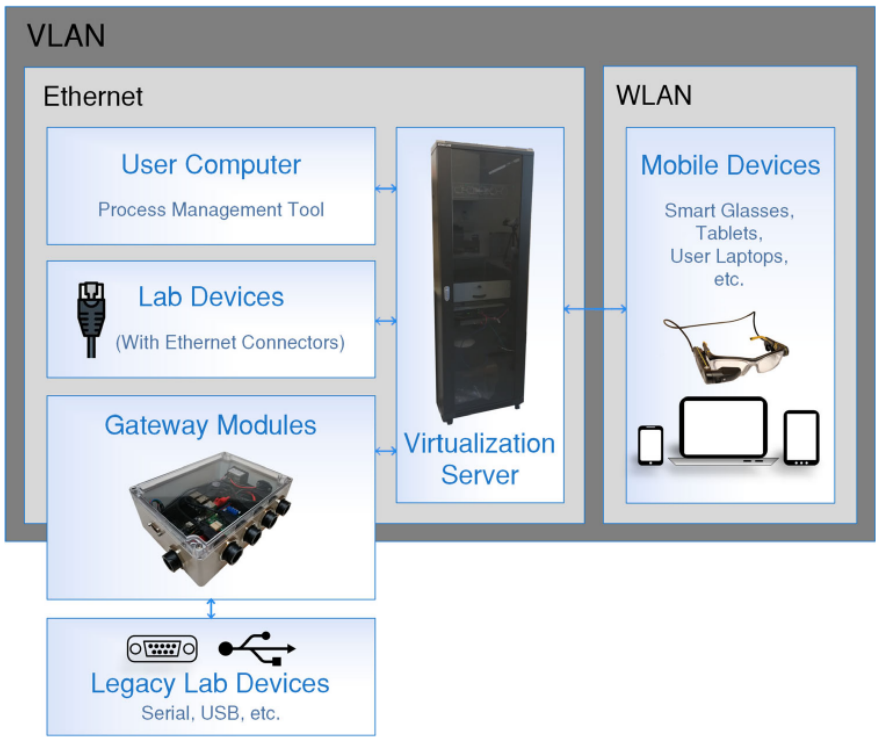


FIGURE 1 Overview of the hardware used in the digitized laboratory. All components are connected to the same VLAN (Virtual Local Area Network) either with wire-based Ethernet connections or to WLAN. Legacy devices are connected to the network using hardware gateway modules. With this network architecture, the flawless communication of all devices is ensured

a single RESTful[3] Hypertext Transfer Protocol web service. All devices are controlled with the SiLA2 protocol.

SiLA2 focuses on consistent communication standards for all devices. It is based on the Google Remote Procedure Call Protocol. All device commands and properties are logically organized in "features." One device can implement several features, which can interact and modify each other if necessary. For example, a magnetic stirrer would implement a "heating," a "stirring," and the mandatory "SiLA2" standard feature. The service, that hosts features and presents them in the network, is called the SiLA2 server. A SiLA2 client can establish a network connection to the server, use the standard feature to obtain documentation on all implemented features and initiate command execution on the device according to this in-place self-documentation.

So far only very few lab devices offer their functionality in form of a SiLA2 server off the shelf. This raises the need for "translators" or gateways between SiLA2 and the vendor-specific protocols. For legacy devices that are connected via the hardware gateway module, the SiLA2 servers for these devices are running on the embedded computer of the module. For every other device, one virtual machine hosts the specific SiLA2 server.

2.3 | Virtualization server

To minimize the demand for hardware and to ease maintenance and scalability a Proxmox Virtual Environment virtualization server is used [14]. Server applications and virtual gateways to lab devices all run as virtual machines on this central virtualization server. One laboratory needs at least one machine for hosting the lab server, one per network-connected device and some more for infrastructure requirements (i.e. controller for WLAN access points, data server(s), etc.).

[3] The term "RESTful" describes an API that follows the principles of Representational State Transfer (REST), which denotes a HTTP-based web API that offers information in a structured way (usually in the format of JavaScript Object Notation (JSON) strings) at pre-defined "endpoints" or URLs (Uniform Resource Locators). The endpoints itself are constant in the server and need to be known *a priori* to the client.

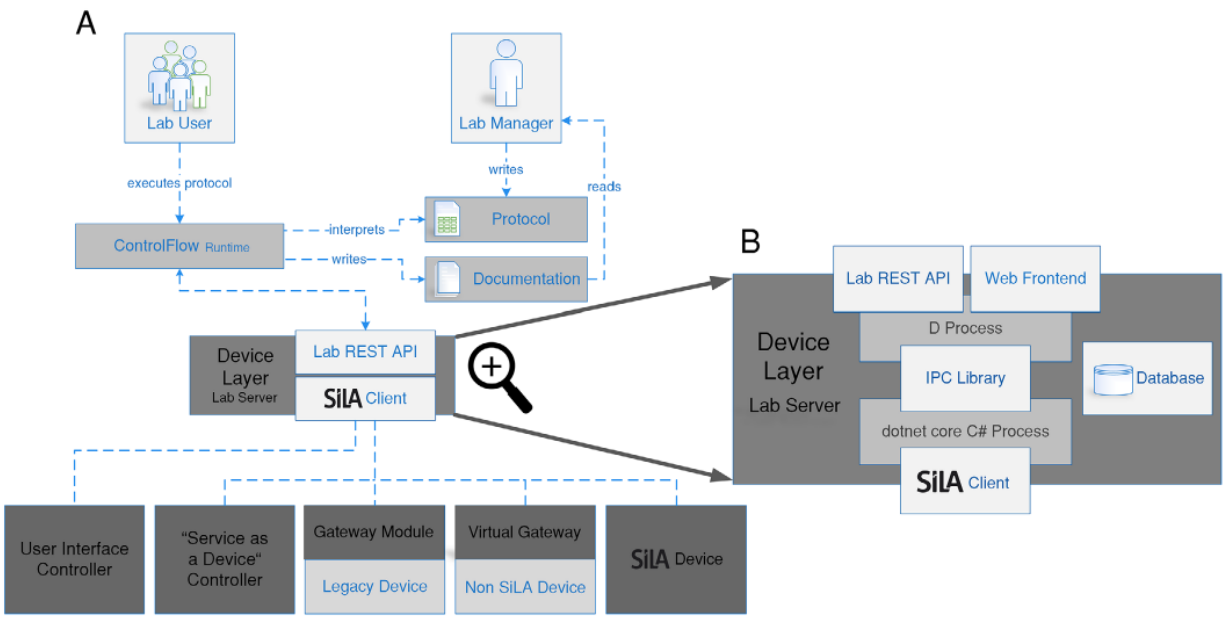


FIGURE 2 The digitized laboratory is based on a central lab server (DeviceLayer). The DeviceLayer implements a generic SiLA2 client that connects all laboratory devices. This is done either directly, via a virtual gateway or using a hardware gateway module. A virtual user interface controller generalizes interaction with UI-devices. Services or resources are implemented as SiLA2 servers and are connected the same way as lab devices. The DeviceLayer also implements a web frontend for administration and a REST API for interacting with the laboratory. All data from the lab is stored in a database that is also running on the same machine as the two DeviceLayer processes

3 | RESULTS AND DISCUSSION

Following a micro-service based strategy the system is implemented as several small components that communicate and interact with strictly defined and standardized interfaces or APIs. In Figure 2A an overview of all soft- and hardware components and their interaction with the human lab manager and researcher is given. The complete system is implemented using free software and the key parts of it were made publically available under appropriate open-source licenses (see Table 1).

All devices are connected to a central laboratory server (DeviceLayer) using the SiLA2 protocol. For devices that do not offer SiLA2 compatibility directly, gateway is in place. This can be either a virtual machine or a physical gateway module. User interface devices are abstracted by a generic UI-device gateway.[4] The DeviceLayer offers a uniform interface for any kind of software to connect to the laboratory. A process control system can use this interface to issue commands on lab devices, collect measurement results and orchestrate steps accordingly. A process management tool ("ControlFlow Runtime"), which is highly focused on user-friendly interaction of the digitized lab with the researcher, was developed in this approach and is described in detail in the supplementary materials.

3.1 | Laboratory server (DeviceLayer)

The central component of the digitized lab is the DeviceLayer laboratory server. One instance of the DeviceLayer forms the virtual representation of "a laboratory" in means of all devices and resources used together. Figure 2B shows the implementation principles of the DeviceLayer. All communication is channeled through this component to ensure full control and data integrity. A database stores every event that occurs in the lab. Furthermore, the Device Layer acts as a central interface for administration. A lab manager can monitor the status of all connected devices and their gateways with a single web interface. The source code for this web interface was described before [12] and is publicly available under an open-source license.

The DeviceLayer communicates with all devices and service-providers using SiLA2 and presents the entry point

[4] Implementation details of this generic approach to user interaction and the principles it implies on digital laboratory protocols are described in the supplementary materials.

TABLE 1 Summary of all tools and libraries used for the presented system including references to documentation and download sources

Element	Tools and libraries used	References
Virtualization server	Proxmox Virtual Environment version: 6.1	https://www.proxmox.com/en/proxmox-ve
SiLA2 servers for: • UI controller • "Service as a Device" controller • Gateway modules • Virtual gateways	• sila_tecan SiLA2 Implementation (C#) Git commit used:@45b977b6 • Publish-System (bash) Git commit used: @0c7a5778	• https://gitlab.com/SiLA2/vendors/sila_tecan • https://gitlab.uni-hannover.de/tci-gateway-module/gateway-publish • Detailed descriptions for developing and deploying in [12]
Gateway modules	• Based on Embedded Computer: Hardkernel ODroid C2 Purchased from Hardkernel Co. Ltd. In December 2019 • Ubuntu Linux Version 3.16 from ODroid-wiki (based on Ubuntu 18.04.3, update-state Mai 2020) • sila_tecan SiLA2 Implementation (C#), ported to dotnet core for embedded applications (see Branch "tci-gwm") Git commit used: @45b977b6 • gRPC (forked to work with embedded processor architectures) Git commit used: @0d417d55 based on: @c564d28f in official repository	• https://wiki.odroid.com/odroid-c2/odroid-c2 • https://ubuntu.com/ • https://gitlab.com/SiLA2/vendors/sila_tecan/-/tree/tci-gwm • https://github.com/grpc/grpc • https://gitlab.uni-hannover.de/tci-gateway-module/grpc • Detailed descriptions for building, installing, programming and running in [12]
Virtual gateways	Arch Linux Linux kernel version: 5.0-5.7, tested with updated rolling system in Mai 2020	https://www.archlinux.de/
DeviceLayer lab server	Arch Linux Linux kernel version: 5.0-5.7, tested with updated rolling system in Mai 2020	https://www.archlinux.de/
Generic SiLA2 client in DeviceLayer	DynamicClient from sila_tecan SiLA2 Implementation (C#) Git commit used: @45b977b6	https://gitlab.com/SiLA2/vendors/sila_tecan
IPC library in DeviceLayer	POSIX Message Queues (Utilized In C# and D) Tested with version from Linux 5.0-5.7	https://www.man7.org/linux/man-pages/man7/mq_overview.7.html
REST server and web-frontend in DeviceLayer	vibe.d REST and web application framework (D) D, dmd/phobos version: 2.089 vibe.d version: 0.8.6	https://vibed.org/
Database in DeviceLayer	mongoDB tested with version: 4.0.6	https://www.mongodb.com/
ControlFlow library and runtime	• curl and curl D library (D) Library D, dmd/phobos version: 2.089 Tested with curl version: 7.67 • imgui GUI library (C++ library invoked from D) commit used: @3bde3750 (v1.75 WIP + docking)	• https://curl.haxx.se/ • https://dlang.org/phobos/std_net_curl.html • https://github.com/ocornut/imgui

All components are either based on open-source software or have been previously described and released by the authors under an open-hardware or open-source license [12]

for all user-side software by hosting a REST API in the network that wraps up all device functions.

The DeviceLayer software is implemented as two separate processes that communicate using an inter process communication mechanism based on Portable Operating System Interface message queues. One part of the DeviceLayer is written in dotnet core C#. It implements a generic SiLA2 client. For that, the "DynamicClient" from the open-source sila_tecan library is used. All devices are connected to the DeviceLayer and one instance of that client is generated for every lab device. The other part of the DeviceLayer is written in D and consists of a database API, the RESTful Hypertext Transfer Protocol web API and a web frontend for laboratory administration purposes. For implementing these three components, the open source vibe.d library is used. An open source NoSQL database (mongo DB) is running on the same virtual machine to store all data of every command executed and every result generated in the lab. The database API models the connection for the other components of the DeviceLayer to use. The REST-API offers all device functionality from the whole lab in one place to interact with for a process management tool or any other client. It can also be used to access data from the database.

This API acts as a central entry point to the lab and offers a consistent way of interacting with all kinds of devices

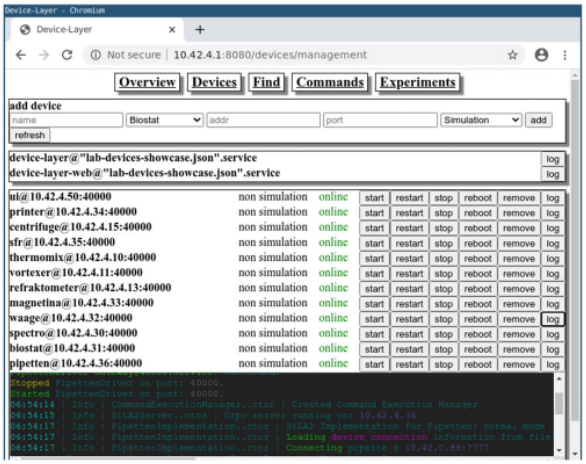


FIGURE 3 Screenshot of the web frontend hosted by the DeviceLayer

and services. When a command or property is requested for a specific device, the availability of the desired device is checked. If the device accepts commands, the SiLA2 call is generated and passed through to the device's SiLA2 server. The command's results are then available for further processing or archiving.

Every command execution is given a unique identifier (ID) by the API and all final or intermediate results are stored in the database. They remain accessible using this specific ID, even when the devices used to carry out the operation seize to work.

The DeviceLayer also hosts a web interface (see Figure 3) that can be used by the lab manager to administrate all laboratory devices. The devices states can be monitored and gateways can be remote controlled. In addition, logs from all SiLA2 servers in the lab are available. SiLA2 devices can be attached to or removed from the lab during runtime of the DeviceLayer.

3.2 | Lab device integration with SiLA2

The SiLA2 Protocol is used for all device communication from the DeviceLayer to lab devices. This means, every device is hosting a SiLA2 server that presents all device functionality on the network. This ensures a streamlined design of the DeviceLayer and encapsulates device integration tasks in small subunits that do not interact with each other. In case several devices of the same type are in use or several devices of the same manufacturer "speak the same language," these units can be duplicated and scaled easily in a horizontal way.

A generic user interface controller, which is also implemented as a SiLA2 server, generalizes interaction with UI-devices (like tablets or smart-glasses). Services or resources are wrapped by SiLA2 servers and are connected the same way as lab devices.

3.2.1 | SiLA2 virtual gateways and gateway modules

Today most devices do not come with a standardized interface and almost no devices offer SiLA2 capabilities. This raises the need for "translators" or gateways from manufacturer/device-specific protocols to SiLA2. Depending on each device-type, either a virtual gateway or a gateway module is used. Every gateway is a standalone application that hosts a SiLA2 server and transforms SiLA2 commands to device specific calls. These are transported depending on the hardware interface of the device. Many new devices offer common interfaces to their devices, such as REST, Extensible Markup Language- Remote Procedure Call or JavaScript Object Notation. Nevertheless, often these services fail in following the chosen standard fully and some extra work has to be done to circumvent this. These devices can be integrated in the lab network directly and a translator gateway can interact with these devices via Transmission Control Protocol/Internet Protocol-communication. Thus, a virtual machine on the virtualization server can run one or more gateways for Ethernet-capable devices. In the solution presented here, for reasons of conformity and scalability, every virtual gateway runs in its own virtual machine.

For legacy devices that only come with a serial- or USB-connector, hardware gateway modules are used. These modules essentially consist of an embedded computer that is connected to the lab device and runs the SiLA2 for that device. The module itself is connected to the lab network via Ethernet and thus enables the flexible integration of older or simpler devices.

All gateways are implemented in dotnet core C# using the open source sila_tecan library. sila_tecan was ported to dotnet core, which allows using the same software on the gateway module and virtual gateways. sila_tecan and also the dotnet core port are available under an open-source license. The usage of the ported variant and also the modifications necessary to use it on the embedded hardware of the gateway module were described by the authors in detail [12]. The gateway modules run Ubuntu/Debian Linux systems, whereas the virtual gateways are running Arch Linux systems. For devices that need manufacturer software that runs uniquely on Windows, the platform independent dotnet core runtime can be used in the same way. The gateway module was released by the authors of this article under an open source/open hardware license [12] and SiLA2 driver development with dotnet core

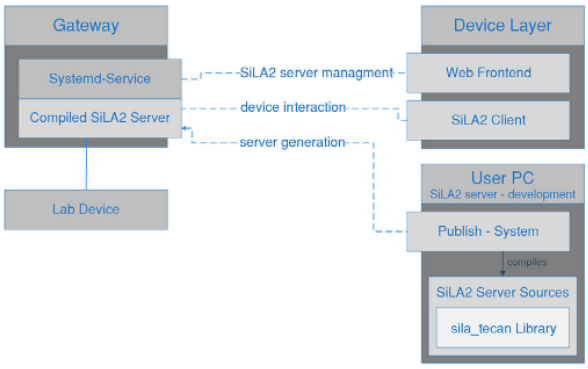


FIGURE 4 Workflow for SiLA2 server development and usage of the publish system and method of remote control of the SiLA2 servers from the DeviceLayer administration web frontend

C# using the sila_tecan library was also described in a step-by-step tutorial [12].

3.2.2 | Development and maintenance of SiLA2 servers in a distributed system

To simplify the development, testing and deployment workflow, a publish system was implemented. Figure 4 visualizes the workflow of the publish system. The developer implements the SiLA2 server functionality on a desktop computer using dotnet core C# and the sila_tecan library. For performance reasons, the publish system precompiles the dotnet core source code to Common Intermediate Language code that is platform independent and can be executed by a dotnet core runtime on the gateway modules or the virtual gateways. The publish system then transfers this software package via Secure Shell to the target gateway and registers a systemd/systemcontrol service for remote controlling the SiLA2 server. This service is controlled by the lab manager via the DeviceLayer's web frontend. The publish-system was described in detail before [12] and is available under an open-source license. It is based on bash-scripts, which were extended for this application by a method for a configuration-file based "batch-deployment," that can update or reinstall the whole lab at once.

3.2.3 | Services and resources as devices

Following the "divide and conquer" principle, non-device services were also wrapped in SiLA2 servers. These are integrated into the digitized laboratory via a virtual gateway in the same way, as actual devices would be. This approach generalizes knowledge and keeps the protocols thin. For example, a complex mathematical computation, such as a process modeling based parameter estimation, becomes a "device" that has a SiLA2 interface. It can be used from within the protocol script via the Lab REST API in the same way as any other lab device.

In a similar fashion, other services, such as automated image analysis tools or an AniML (Analytical Information Markup Language) generator can be integrated. In addition, a resource planner or scheduler can be added. When there are multiple devices of the same type in the laboratory, a "meta-device" for scheduling would implement the same SiLA2 interface as the devices themselves. The protocol would only interact with that scheduler and the task would automatically be performed on the first device from the pool becoming available.

3.3 | Practical example for the setup and execution of a digitized workflow

To increase understanding, an example workflow for water analysis is now discussed. The focus lies on highlighting the principle steps that need to be taken to encounter a digital transformation. Due to reasons of complexity, this cannot be a detailed step-by-step tutorial. However, the systematics for SiLA2 gateways and its central administration, the implementation of SiLA2 servers, the usage of the publish system and the underlying principles for system and software design are all available under open-source licenses. They were described in detail (including step-by-step bash command instructions and extensive source code documentation) in an earlier publication [12].

3.3.1 | Technical requirements and protocol description

As a starting point it is assumed, that the hardware and infrastructure is set up as described in the Materials and Methods section. A local area network that can connect all components is in place. A laboratory server running the DeviceLayer is accessible in this network. The DeviceLayer is the central entry point for administration of all lab devices (web-frontend), process management (REST-API) and data storage (database). A hardware gateway module has been build and installed following the instructions in [12].

The schematics of the water analysis workflow are depicted in Figure 5. A given water sample should be analyzed for microbial contamination. For that, a defined volume of the sample is first transferred into a digestion buffer. The digestion is carried out in a combined heating

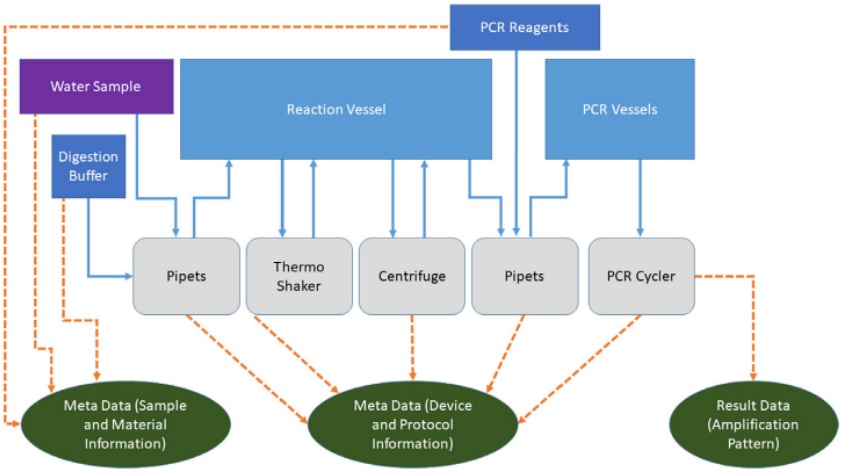


FIGURE 5 Schema of the exemplary water analysis workflow. The sample is represented by the purple box, all inventoried materials by the dark blue boxes. Light blue boxes: reaction vessels; Grey boxes: lab devices. Data generated by the process is displayed in the green cycles. Solid blue arrows: physical transfer of materials or vessels; Dashed orange arrows: data generated in specific locations

and shaking device (thermo-shaker). After the digestion, the cell debris is separated by centrifugation and the supernatant is used for a Polymerase Chain Reaction (PCR) based detection of microbial genetic material.

For this rather simple workflow, four lab devices are needed: pipettes, thermo-shaker, centrifuge and PCR cycler. It is assumed, that all devices offer some kind of digital interface to allow remote control and data aggregation. The pipettes are connectable to a WLAN network and are controlled by a JavaScript Object Notation-Remote Procedure Call protocol. The thermo-shaker has a Recommended Standard 232 serial interface and can be controlled by a simple serial protocol. The centrifuge has a USB connector, which is mapped internally to a USB-serial converter. Thus, this device is also controlled by a serial protocol. The PCR cycler has an Ethernet connector and runs a REST server that offers endpoints to control the device. This example situation reassembles a common starting point in lab digitization.

Additionally smart-glasses offer the possibility to guide the user through the process and provide additional information. The existence of an inventory system is assumed, that is used for chemical registration. All chemical containers are registered with this system and are identifiable via a ID that is printed on the containers as a Quick Response code. The inventory system offers a REST-API to access all information regarding a special ID.

3.3.2 | Device integration and workflow setup

To connect the devices needed for this workflow to the laboratory server, they first need to be placed in the network. For the pipettes and the smart-glasses, this means connecting them to a WLAN access point in the laboratory network. The PCR cycler is plugged into the same network using an Ethernet cable directly. The centrifuge (USB-connector) and thermo-shaker (serial-connector) are plugged into the gateway module which itself is connected to the lab network using its Ethernet port.

After these physical requirements are met, all devices need to be linked to the laboratory server. This means that for every device a SiLA2 server needs to be running in the network that translates SiLA2 calls to the device specific protocol. For the pipettes and the PCR cycler virtual machines on the virtualization server are set up to run those servers. For the centrifuge and thermo-shaker, the SiLA2 servers are to be run on the gateway module. The developer now needs to implement all SiLA2 servers following the micro-service template pattern that is used by the publish system. This essentially means describing and documenting the SiLA2 interface and wrapping up the device specific calls described in the devices manual into the SiLA2 structure. The implementation of a SiLA2 server using the sila_tecan library and dotnet core C# is documented in detail in [12] and in the "SampleServer" in the sila_tecan repository. Also the special requirements (using a Google Remote Procedure Call implementation modified for embedded processor architectures) outlined in [12] needs to be taken into account when developing servers for the gateway module.

When the SiLA2 servers are available and were compiled and deployed to the target machines (refer to [12] for detailed instructions) they can be added to the DeviceLayer using the web-frontend.

For the user interface on the smart-glasses, the generic UI SILA2 server abstracts the communication to the

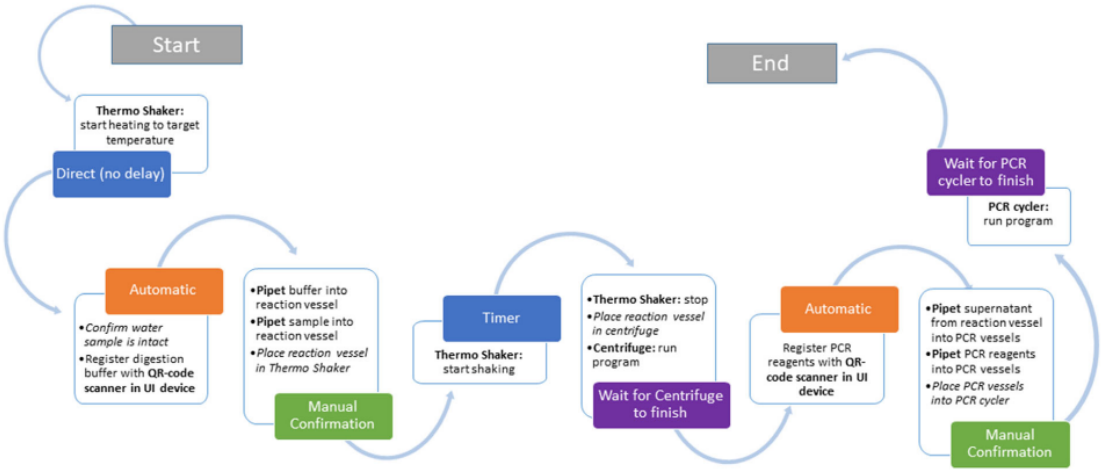


FIGURE 6 Simplified overview of the steps and transitions necessary for the water analysis example workflow. White boxes represent step content. Bold font: a device is utilized; Italic font: manual tasks with no direct digital interaction. The colored boxes attached to the steps represent the transition conditions that need to be fulfilled for leaving a step. Blue: transitions with no direct interaction with the lab or the researcher (timers or direct transitions); Orange: transitions, which occur as soon as the last task inside the step is performed; Green: transitions require the manual confirmation of the researcher (pushing a button or using a speech command on a UI device); Purple: transitions that wait for a device command to finish

DeviceLayer.[5] Following the "Services as Devices" pattern, the inventory application is wrapped in a SiLA2 server that also runs on a virtual machine. If the results of the experiment should be available according to the FAIR data principles [8], a generator, that takes all experimental data and metadata and generates an AnIML file, which it uploads to a publically available repository, can be integrated in the same way. This enables the usage of the system for example in context of the NFDI. When a repository for research data are available (either locally or in the internet), these standardized data can be uploaded to it automatically by the AnIML-Generator or a specialized micro-service.

After adding the SiLA2 servers to the device list in the DeviceLayer, all necessary functionality of the lab can be accessed using its REST API. To set up a workflow, a process management tool needs to be connected to this API. For this example the ControlFlow library is used, that is described in the supplementary information. Workflows are described in a script-like format and are made up of steps and transitions between them.

For this example workflow, the simplified protocol layout is shown in Figure 6. The corresponding ControlFlow script can be found in the supplementary information. The script is compiled to a ControlFlow Runtime, which can be executed by the lab user on any computer connected to the lab network.

[5] The mechanisms of abstraction that are used to distribute device runtime information to all UI and its implications on protocol design are discussed in the supplementary material.

When starting a run of this ControlFlow Runtime a unique experiment ID is automatically created in the DeviceLayer database. All experimental data generated during this run is automatically linked with this ID. This creates structural context of result data from different devices and links this with additional metadata about the experimental setup. For accessing the DeviceLayer database, the REST-API can be used by any data analysis tool (either directly or via a connecting middleware). For quick database queries, the DeviceManager web frontend of the DeviceLayer offers a graphical tool for viewing experimental data.

3.3.3 | Execution of the digitized workflow

After all necessary devices and the workflow script for the process control system are in place, the new digitized workflow can be started. During the process, many different kinds of information are available and are presented by the system at different levels of abstraction.

The lab user is presented on the smart-glasses with screens that display short information snippets about the current step of the protocol or context sensitive information. During the pipetting steps the materials to use are shown together with relevant safety information, whereas during the denaturation or centrifugation step, the remaining runtime of the device is displayed. The smart-glasses are also used to scan the Quick Response-codes generated

by the inventory system. The contents of these codes are fed into the "virtual" inventory device. This enables runtime checking for correctly used chemicals and expiration dates. It also enables the inventory system to track the use of all registered materials automatically.

The ControlFlow executable presents the user with a graphical user interface that allows process coordination (jumping between steps, re-running steps, etc.) and shows the communication between devices and lab server in a condensed form. The lab manager can use the DeviceLayer web-frontend to view the logs from all SiLA2 servers in the laboratory in one place. This can be used for validating processes or troubleshooting and testing new workflows.

During the experiment all data generated by the lab is stored directly in the database of the DeviceLayer. From there it is instantly accessible for the process management system (for example to determine the remaining runtime of a device or intermediate measurement results, etc.) but is also archived for later use. When a run of the workflow is started, the lab server generates an ID that ties all pieces of information and results generated in this run together.

The REST-API of the DeviceLayer can also be used to feed experimental data into an electronic laboratory notebook (i.e. by using an appropriate middleware that "speaks" both APIs) or any other data analysis tool for further processing.

4 | CONCLUDING REMARKS

In this report, a detailed concept for laboratory digitization is presented. The methodology used focuses on micro-services, which have clearly defined purposes and interfaces. It thus allows easy maintenance and extension. The data flow is modeled in a strategic way, which ensures data validity and enables easy validation and certification of processes. In contrast to other approaches, which were focused on fully automated applications, the method presented aims at a broad digital integration and flawless interaction with the human worker in the laboratory.

Wherever possible, existing technologies, protocols and standards were used. However, this often raises the need for integration of commercial components with specific communication protocols. In the future, when more and more digital interaction with lab devices will be required, the demand for a vendor independent common communication protocol will grow. For implementing standards like the FAIR-data principles, an easy method of data acquisition and processing is without alternatives. This is enforced for example by the German NFDI and systems as the one presented in this article will increase data quality and reduce the need for potentially error-prone manual data processing by the researcher.

Especially in research or quality testing, were handwritten protocols that are carried out by trained professionals are the standard workflow, a sophisticated method for interaction of the digitized lab with the human worker is necessary. The approach presented focuses on good adoption through easy to use hardware and software that does not interfere with the normal lab process. A great improvement over traditional methods are the possibilities for fully automated documenting and archiving. As all data generated in the lab process are available in one place, creation of analysis protocols can be automated to exploit the full potential of the data.

ACKNOWLEDGMENTS

The authors thank the German Federal Ministry of Education and Research for funding this research in context of the project "Digitalisierung in der Industriellen Biotechnologie (DigInBio)" (BMBF FKZ 031B0463C) and the Projektträger Jülich.

The authors also thank their partners in that project: the Institut für Bio- und Geowissenschaften, Department for Bioprocesses and Bioanalytics at the Forschungszentrum Jülich and the Faculty of Mechanical Engineering, Institute of Biochemical Engineering at the Technical University of Munich.

They also thank the Tecan Software Competence Center for help and support with their open source SiLA2 implementation (sila_tecan).

Open access funding enabled and organized by Projekt DEAL.

CONFLICT OF INTEREST

The authors have declared no conflict of interest.

DATA AVAILABILITY STATEMENT

The data that support the findings of this study are available from the corresponding author upon reasonable request.

ORCID

Marc Porr https://orcid.org/0000-0002-5280-7282

REFERENCES

1. Bär, H., Syré, U., Infoteam sila library simplifies device integration. J. Lab. Autom. 2011, 16, 371–376.
2. Austerjost, J., Bargholz, M., Porr, M., Marquard, D. et al., A flexible IT infrastructure for the integration of smartglasses into the brewing laboratory as a digital support for standard analysis workflows. BrewingScience 2019, 72, 1–9.

3. Schmid, I., Aschoff, J., A scalable software framework for data integration in bioprocess development. Eng. Life Sci. 2017, 17, 1159–1165.
4. Lütjohann, D.S., Jung, N., Bräse, S., Open source life science automation: design of experiments and data acquisition via "dial-a-device." Chemom. Intell. Lab. Syst. 2015, 144, 100–107.
5. Delaney, N.F., Echenique, J.I.R., Marx, C.J., Clarity: an open-source manager for laboratory automation. J. Lab. Autom. 2013, 18, 171–177.
6. Chapman, T., Lab automation and robotics: automation on the move. Nature 2003, 421, 661–666.
7. Frey, J.G., Dark lab or smart lab: the challenges for 21st century laboratory software. Org. Process Res. Dev. 2004, 8, 1024–1035.
8. Boeckhout, M., Zielhuis, G.A., Bredenoord, A.L., The FAIR guiding principles for data stewardship: fair enough? Eur. J. Hum. Genet. 2018, 26, 931–936.
9. Blow, N., Lab automation: tales along the road to automation. Nat. Methods 2008, 5, 109–112.
10. Bär, H., Hochstrasser, R., Papenfuß, B., SiLA: Basic standards for rapid integration in laboratory automation. J. Lab. Autom. 2012, 17, 86–95.
11. Gauglitz, G., Lab 4.0: SiLA or OPC UA. Anal. Bioanal. Chem. 2018, 410, 5093–5094.
12. Porr, M., Schwarz, S., Lange, F., Niemeyer, L. et al., Bringing IoT to the lab: sila2 and open-source-powered gateway module for integrating legacy devices into the digital laboratory. HardwareX 2020, e00118.
13. Austerjost, J., Porr, M., Riedel, N., Geier, D. et al., Introducing a virtual assistant to the lab: a voice user interface for the intuitive control of laboratory instruments. SLAS Technol. 2018, 23, 476–482.
14. Daniels, J., Server virtualization architecture and implementation. XRDS Crossroads, ACM Mag. Students 2009, 16, 8–12.

SUPPORTING INFORMATION

Additional supporting information may be found online in the Supporting Information section at the end of the article.

How to cite this article: Porr M, Lange F, Marquard D, et al. Implementing a digital infrastructure for the lab using a central laboratory server and the SiLA2 communication standard. *Eng Life Sci*. 2021;21:208–219. https://doi.org/10.1002/elsc.202000053

3.6.1 UNTERSTÜTZENDE INFORMATIONEN

Supplementary Information for the Article

"Implementing a Digital Infrastructure for the Lab Using a Central Laboratory Server and the SiLA2 Communication Standard"

Marc Porr, Ferdinand Lange, Daniel Marquard, Laura Niemeyer, Patrick Lindner, Thomas Scheper, Sascha Beutel

A – Process Management in the Digitized Lab

Methods or protocols are "run on the lab" by a process management tool. This connects to the REST-API of the laboratory server, runs commands and interprets results. The DeviceLayer is agnostic of the process management tool used, so in principle any method desired to connect a protocol manager (if necessary by utilizing an appropriate middleware) can be used.

In the solution presented, a process management tool that specifically targets good interaction of the digitized laboratory with the human researches was developed, especially to be used in context with the DeviceLayer.

With this tool standard operation procedures (SOPs) are defined in an easy to learn script-like format. Steps and transitions are defined and calculations or conditions can be implemented in a flexible way. This layout opens up the possibility to extend the infrastructure by a graphical tool for process design later on.

A.1 – Implementation of the Process Management Tool (ControlFlow)

The process management tool, called "ControlFlow", interacts with the whole lab in one place via the Lab REST API of the DeviceLayer. The ControlFlow runtime can run on any user's computer in the network. It sends command requests to the DeviceLayer and receives result data and device state information from it.

Figure 1 visualizes the architecture of the ControlFlow runtime in context with specific procedures. Procedures and the steps they contain are interpreted by the runtime and all device commands are cached in a command storage. Corresponding calls to the Lab REST API of the DeviceLayer are send and every change of state or (intermediate) result is cached back to the command storage. Timers are running locally in the runtime. A GUI component can be used by the worker in the lab to view the protocol and the current state of step execution.

ControlFlow is implemented as a library in the D programming language. For protocol design, this library offers convenience functions to wrap up calls the DeviceLayer. Protocols are written in a mixin format, so they essentially form "compile time plugins" for the ControlFlow main program. This program presents the lab user with a graphical user interface (GUI) and mechanisms of controlling a running protocol.

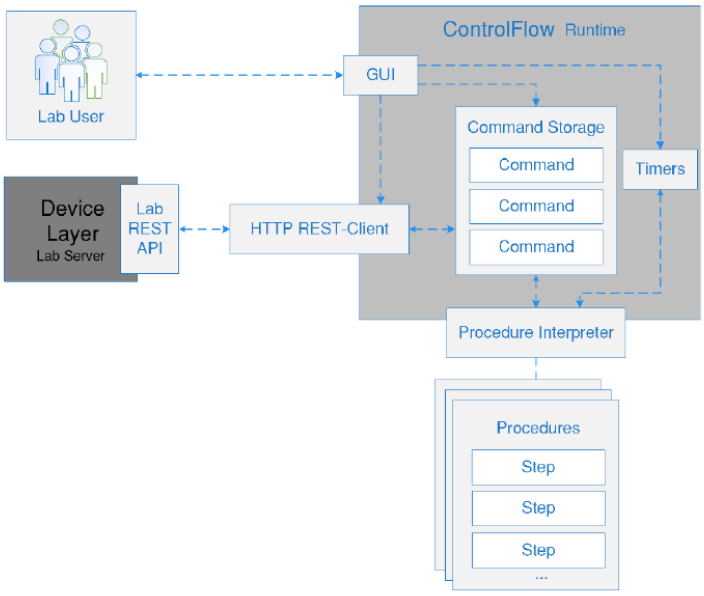


Figure 1: Architectural overview of the ControlFlow implementation.

The protocol mixins are compiled together with the main component to a ControlFlow Runtime. This executable offers all protocols that were beforehand defined. This method ensures that somebody executing a protocol must not necessarily have the permissions to alter its steps. Compiling protocols into the executable offers precise control over user rights. The protocol designer (lab manager) can pre-define the steps and this can only be changed by the lab user, when he/she is given the corresponding source code.

To introduce a precisely defined level of flexibility, a parameter file bases approach is taken. Parameters and their boundaries are defines in the protocol itself (and are thus unchangeable in the compiled protocol). However, specific parameter values are defined in a parameter file in JavaScript Object Notation (JSON) format. This file can be changed by the user before executing a protocol. This enables the executor to alter all parameters, which were allowed to be changed by the manager, in pre-defined boundaries. This method ensures that the lab manager has full flexibility during design time, whereas the liberty of adjustment and configuration by the worker in the lab can be specifically fine-tuned.

A.2 – Protocol Design

Protocols are expressed in the form of scripts that consist of steps and transitions between them. Steps usually contain lab device interaction and/or user interaction via the generic user interface (UI) controller. Transitions specify how the current step can be left for the next one. This can either mean a direct transition with no conditions (devices can thus perform tasks in the background), waiting for a device command or timer to finish, or waiting for some user input – like the confirmation of manual

steps (i.e. pipetting, transferring containers or liquids, etc.). To model loops or forks in the control flow, steps can specify the following step based on conditions or logic operations.

As the “script language” for protocol design is developed as a library for the D programming language, all kinds of complex operations (like mathematical calculations, data operations, etc.) can be performed in a protocol. This flexibility ensures that even complicated workflows can be accurately expressed in one place with all process logic combined together. During the planning-phase of an experiment, the lab manager designs and parametrizes the workflow. This protocol can then be evaluated, validated and probably certified. Later on, every worker in the lab can execute the protocol. The workflow can contain gatekeepers, which only allow the worker to proceed, when certain safety or quality criteria are met. For example in a stirring step, the digitized laboratory can make sure that the duration and stirring speed was sufficient by a defined margin of tolerance. The server generates unique identifiers for every resource, which can be used by the ControlFlow runtime to recall all results and measurements any time in the future.

Figure 2 shows an example protocol that can be executed in the digitized lab. The ControlFlow library is used to specify a simple protocol that consists of three steps. Firstly, a storage object is initialized, that hold information about the protocol and metadata about the UI-devices supported.

The first step displays information for the worker on any UI-device connected and waits for a manual confirmation by the worker. The second step starts a centrifuge with 2000 RPM at 20 C° for 60 seconds and waits for the centrifuge to finish this command. Lastly, a timer is set to 25 seconds and is awaited.

```
1  module usecases.example;
2  import controlflow.usecase;
3  mixin RegisterAllUseCases;
4
5  @UseCase void example()
6  {
7      flow.initStorage([UIDev.glass], "TCI", "example", "layouts/example");
8
9      flow.step("prepare", {
10         sendLayout("prepareCentrifugeUI");
11         uiEvents("next").wait();
12     });
13
14     flow.step("centrifuge", {
15         auto cmd = command("centrifuge", "Start", 2000, 60, 20);
16         cmd.wait();
17     });
18
19     flow.step("timer", {
20         sendLayout("timerUI");
21         wait(timer("wait", 25.seconds));
22     });
23 }
```

Figure 2: Example protocol file.

In a protocol, several steps are specified. A step is implemented as a function delegate that can contain any valid D source code. The ControlFlow library offers functionality for device interaction by wrapping the client for the DeviceLayer's Lab REST API. In addition, an abstraction layer for parameters and a generic interface to UI-device interaction methods is available to the protocol designer. When a step delegate is finished, the concurring step is automatically entered by the runtime. This transition can be delayed by waiting for certain tasks. For example, the step can wait for a timer or for a device command to finish. This is necessary, when the command's results are used in the next step. For manual steps, where no device is involved, waiting for an event trigger from an UI-device is necessary. For complex control logic, conditional jumps between steps can be used. Thus, forks, loops and conditional execution of steps can be modeled.

A.3 – Protocol Execution

After a protocol is packed as an executable ControlFlow Runtime, it can be executed on any user computer that has network access to the laboratory server. When starting the compiled ControlFlow executable with a protocol, the user is presented with a graphical user interface (GUI), which allows managing the protocol's steps and visualizes the data transferred between the devices, the lab server and the process management tool (see Figure 3). On the top right all steps in the protocol can be seen and control flow can be manipulated (repeat steps, jump to steps, pause execution, etc.). On the top left calls to the Lab REST API are listed. For debugging purposes payload can be seen and resending of commands is possible. On the bottom, execution logs from the protocol runtime mechanism are printed to inform the user or lab manager of the precise interaction of the process management with the lab server.

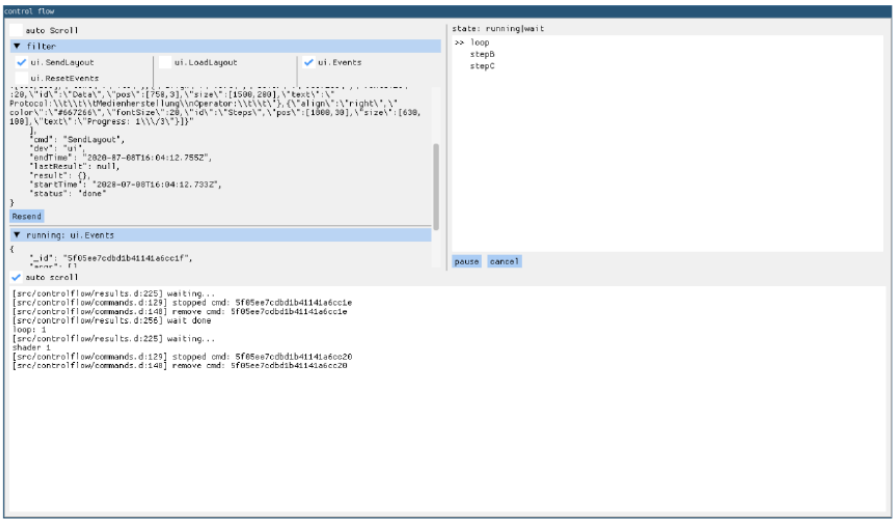


Figure 3: Screenshot of the ControlFlow GUI during execution of a simple dummy protocol.

B – Guiding and Aiding the User during Protocol Execution

For communication with the worker during protocol execution, different user interface devices (UI-devices) were integrated. These devices offer guidance to the worker in form of short summaries of the current protocol step, safety information or relevant data about the chemical/organism/material worked with. This can also include the results of automatic calculations. For example, when a cell density measurement is performed, the result value is automatically processed with the calibration stored in the protocol and the worker is presented with the resulting cell concentration directly.

The communication of the laboratory system with the researcher during the procedure is modeled using a standard gateway to all possible UI-devices. Thus, a procedure can be agnostic of the UI-device used. It only needs to specify what needs to be presented to the user, not how or by which channel the communication to an end-device is carried out. All used UI devices are connected to the lab server in a standardized way and receive the information necessary to equip the worker with all information specified in the protocol.

In the presented solution, user interaction with two different kinds of UI-devices is implemented. One is based on a consumer-grade tablet that is carried by the worker and placed on the lab bench when a protocol is carried out. The tablet presents visual information and offers buttons for protocol navigation. The other approach uses smart-glasses that are worn by the worker on top of the lab safety glasses. They are a head-mounted display that shows small snippets of visual information right in the view of the worker. Controlling the protocol execution is achieved in this setting by simple voice commands like “confirm”, “previous”, etc. However, due to the generic nature of the mechanism, any other device can be plugged into the architecture without changing the principles of data flow or the protocols.

B.1 – User Interaction Abstracted by a Generic User Interface Device

Communication with different UI-devices is abstracted by a generic UI controller. It is implemented in the same way as any other SiLA2 server. The uniform and generic SiLA2 interface of this controller is consumed by the DeviceLayer and it thus offers the UI functionality in the same way as it is done with any other device’s commands by the REST API for the process management tool. This generic device can be used during protocol design. Depending on the UI-devices connected during protocol execution, the content is formatted according to the specific target device.

Figure 4 shows the architecture for integrating UI-devices into the digitized laboratory. Every target device has to run a small REST server that hosts an API, which supports receiving of layout information for display purposes and sending of events. This, for example, is done with an Android app on the tablets and smart-glasses. Other UI-devices can be integrated the same way, as long as the common UI REST API is shared. The generic UI controller connects to all of these UI REST servers of all UI-devices present.

Whenever new information is announced by the ControlFlow runtime, this information (in JSON format) is pre-processed by the UI controller and is sent to all UI-devices connected.

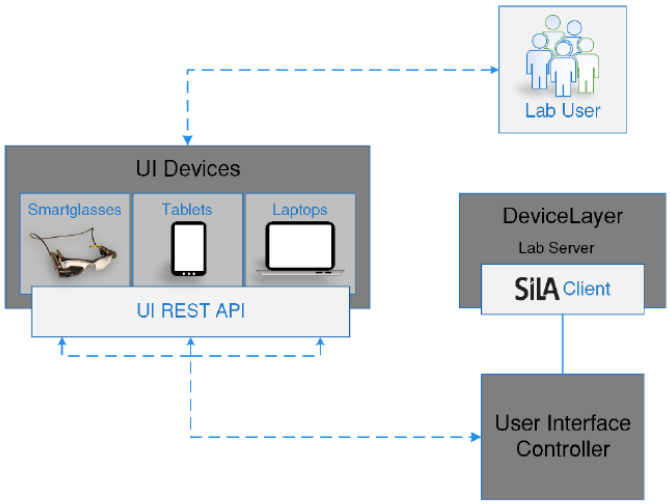


Figure 4: Architecture of UI-device integration via the generic user interface controller (SiLA2 server).

When a user interaction event (such as the trigger for a manual confirmation of a step using speech commands or by pressing buttons on the UI-device) is registered on one of the connected UI-devices, this information is passed on via the UI REST API to the generic UI controller. The controller sends it via the DeviceLayer REST API to the ControlFlow runtime. There it is automatically processed and the runtime decides, whether this event is further processed at the current protocol execution state. If, for example, the runtime is waiting for the user to manually confirm a transfer step, the following step is entered after the "confirm"-event was registered by one of the connected UI-devices.

B.2 – Flexible Layout Files Specify Content

The process management tool uses JSON data, which is send via the REST API of the DeviceLayer to the generic UI controller, to specify the information that should be presented to the researcher. The Apps running on the specific UI devices interpret the JSON content and generate corresponding UIs. This means, that every device can for example calculate image sizes or text box positions based on its available screen size.

To simplify protocol design, templates for common visual elements are stored in JSON files and are loaded by the ControlFlow runtime during execution. These files can contain variables, which can be filled in by using a variable substitution mechanism in the ControlFlow library. This method ensures that only minimal redundant UI specification is needed inside the protocols.

To display life measurement data, placeholder fields are specified inside the JSON layouts. Every time a new (intermediate) result is generated inside the lab, the DeviceLayer sends an update to the generic UI device controller. It then passes on this information to all connected UI devices and the Apps on these devices can update visual information wherever the new updated field is displayed. In this mechanism, the source devices of results are not described by their specific (unique) IDs, but by their

class. This makes devices interchangeable (for example swapping a scale from manufacturer A for one of manufacturer B) without changing the protocols or layout files.

B.3 – Example

Figure 5 displays an example screen, shown to the worker on the smart glasses during a weighing protocol step that is part of a media composition protocol for a bioreactor. The view is updated with the current scale reading in real time during the weighting process. It also shows the desired target weight. The information is displayed right in the field of view of the lab worker, who can work with both hands. The protocol execution waits as long as it takes to weigh in the target weight defined in the protocol by a specific margin of tolerance. This makes sure that no mistake can be made in this step. The exact amount of material is stored in the database after the step is finished and can be used for precise calculation of concentrations or for documentation purposes.

Figure 6 shows the ControlFlow script needed for this example step. The layout template is stored under the name “4_Weight” and contains a variable field named “chemical”. To make use of this template, first the variable “chemical” has to be set to the name of the material used in this step (“yeast exract”) using the “flow.storage.set”-command in line 2. With the “sendLayout”-command in line 3the template is loaded and send to the generic UI controller, which results in the generation of the screen displayed in Figure 5.

Line 4 contains the device command to the scale. It calls the SiLA2 command “WeightTotargetValue” which needs two parameters. The first is the target Value (5g in this case) and the second is the margin of tolerance that is considered acceptable. The “waitUntilDone” command acts as a gatekeeper and makes sure that the step can only be left for the following one, when the weighting command reports a mass inside the defined range.

The used layout template “4_Weight” is shown in Figure 7. It describes the format of the information to display in a JSON structure. The “mode” field describes the type of events that the UI device needs to listen. As this step should only be left by performing the weighting task, this is set to “none”.

Figure 5: Example view of the content presented on the smart glasses to the lab worker during a weighting step that is part of a media composition protocol for a bioreactor.

```
flow.step("Weigh Yeast Extract", {
    flow.storage.set("chemical", "yeast extract");
    sendLayout("4_Weight");
    command("Scale", "WeightToTargetValue", 5, 0.1).waitUntilDone();
});
```

Figure 6: ControlFlow script for the example step described above

The layout defines one image and three text boxes. One shows the descriptive text, which also contains the variable “chemical” which is overwritten by the value used in the ControlFlow script. This architecture makes layout templates flexible and reusable and thus minimizes repetitive declarations. The next two text boxes do not contain fixed values. They are linked to SiLA2 result fields of the Scale. When the command to the scale is issued and the “WeightToTargetValue” command reports intermediate results to the DeviceLayer, these fields are dynamically updated.

```
{
    "mode": "none",
    "textBoxes": [
        {
            "pos": [ 45, 30 ],
            "text": "Please weigh in $(chemical)"
        },
        {
            "pos": [ 45, 60 ],
            "id": "Scale:CurrentValue"
        },
        {
            "pos": [ 45, 80 ],
            "id": "Scale:TargetValue"
        }
    ],
    "images": [
        {
            "pos": [ 15, 30 ],
            "img": "picto/BeakerScaleAdd.png"
        }
    ]
}
```

Figure 7: Simplified example layout template that describes the UI information used to create the UI screen shown in the example above.

C – ControlFlow script for Water Analysis Example Workflow

The following Listing contains the ControlFlow script for the PCR based water analysis workflow described in the example section of the article.

```
module usecases.waterAnalysis;

import controlflow.usecase;
import std.json;
import std.file;

mixin RegisterAllUseCases;

@UseCase void waterAnalysis()
{
    // read Parameters from config-file
    JSONValue config = parseJSON(readText("params/waterAnalysis.json"));

    // create UI Helper
    flow.initStorage([UIDev.tablet, UIDev.glass], "TCI", "PCR Water Analysis",
  "layouts/waterAnalysis");


    // setup the workflow:

    flow.step("Login with personal QR-code", {
        flow.storage.set("name", "PCR Water Analysis");
        sendLayout("Login");
        registerUser(waitForQrCode());
    });

    flow.step("Heat Thermo Shaker", {
        command("thermoshaker", "Heat", config.opt!int("Temperature"));
    });

    flow.step("Register Digestion Buffer", {
        string chemname = "Digestion Buffer";
        flow.storage.set("chemical", chemname);
        sendLayout("Scan");
        registerChemical("digestion_buffer", chemname);
    });

    flow.step("Pipet Digestion Buffer", {
        flow.storage.set("chemical", "Digestion Buffer");
        flow.storage.set("target", "Reaction Vessel");
        sendLayout("Pipet");
        command("pipet", "SetVolume", config.opt!float("BufferVolume"));
        wait(uiEvents.Confirm);
    });

    flow.step("Pipet Water Sample", {
        flow.storage.set("chemical", "Water Sample");
        flow.storage.set("target", "Reaction Vessel");
        sendLayout("Pipet");
        command("pipet", "SetVolume", config.opt!float("SampleVolume"));
        wait(uiEvents.Confirm);
    });

    flow.step("Transfer to ThermoShaker", {
        flow.storage.set("message", "Place Reaction Vessel in Thermo Shaker");
        sendLayout("Message");
        wait(uiEvents.Confirm);
    });

    flow.step("Digest", {
        flow.storage.set("message", "Please wait for Digestion");
        sendLayout("Wait");
        auto timeout = timer(5.minutes);
        wait(timeout);
    });

    flow.step("Stop Thermo Shaker", {
        command("thermoshaker", "Stop");
    });

```

```
70.     flow.step("Transfer to Centrifuge", {
71.         flow.storage.set("message", "Take Reaction Vessel out of Thermo Shaker, place
    it in Centrifuge. Place Counter Weight.");
72.         sendLayout("Message");
73.         wait(uiEvents.Confirm);
74.     });
75.
76.     flow.step("Run Centrifuge", {
77.         sendLayout("CentrifugeRun");
78.         auto cmd = command("centrifuge", "Run", config.opt!int("CentriRPM"),
   config.opt!int("CentriRuntime"));
79.         cmd.waitUntilDone();
80.     });
81.
82.     flow.step("Transfer out of Centrifuge", {
83.         flow.storage.set("message", "Take Reaction Vessel out of Centrifuge");
84.         sendLayout("Message");
85.         wait(uiEvents.Confirm);
86.     });
87.
88.     flow.step("Register PCR Reagents", {
89.         string chemname = "PCR Master Mix";
90.         flow.storage.set("chemical", chemname);
91.         sendLayout("Scan");
92.         registerChemical("pcr_mastermix", chemname);
93.     });
94.
95.     flow.step("Pipet Sample Supernatent", {
96.         flow.storage.set("chemical", "Supernatent of Water Sample");
97.         flow.storage.set("target", "PCR Vessels");
98.         sendLayout("Pipet");
99.         command("pipet", "SetVolume", config.opt!float("SupernatantVolume"));
100.          wait(uiEvents.Confirm);
101.      });
102.
103.      flow.step("Pipet PCR Reagents", {
104.          flow.storage.set("chemical", "PCR Reagents");
105.          flow.storage.set("target", "PCR Vessels");
106.          sendLayout("Pipet");
107.          command("pipet", "SetVolume", config.opt!float("PCRReagentsVolume"));
108.          wait(uiEvents.Confirm);
109.      });
110.
111.      flow.step("Transfer into PCR Cycler", {
112.          flow.storage.set("message", "Transfer PCR Vessels into PCR Cycler");
113.          sendLayout("Message");
114.          wait(uiEvents.Confirm);
115.      });
116.
117.      flow.step("Run PCR Cycler", {
118.          sendLayout("PCRRun");
119.          auto cmd = command("pcr", "RunProgram", config.opt!int("PCRProgramName"));
120.          cmd.waitUntilDone();
121.      });
122.
123.      flow.step("End", {
124.          sendLayout("Finish");
125.      });
126.  }
```

3.7 DIGITALE INTEGRATION EINER PERIODIC COUNTER CURRENT CHROMATOGRAPHIE ANLAGE

Da wissenschaftliche Geräte immer komplizierter werden, wird der Einsatz von Geräten der ersten Generation oder *Proof-of-Principle*-Geräten in einem wissenschaftlichen Umfeld immer schwieriger. Hersteller stellen für kauffertige Geräte voll funktionsfähige, grafische Benutzersoftware zur Verfügung, die intuitiv zu bedienen ist und leicht für den täglichen Gebrauch verwendet werden kann. Komplexe Geräte, die in Forschungseinrichtungen gebaut wurden, um neue Methoden oder Prinzipien zu testen, sind in der Regel nicht so intuitiv zugänglich. Sie müssen über textbasierte Schnittstellen (Konfigurationsdateien, Programm-Skripte, Terminal-Befehle) gesteuert werden.

Dies führt dazu, dass selbstentwickelte Lösungen nur selten genutzt werden, nachdem ihre prinzipielle Funktionalität bewiesen wurde. Die Entwicklung maßgeschneiderter grafischer Software für solche Zwecke ist jedoch arbeitsintensiv und erfordert einen hohen (nicht direkt wissenschaftlich relevanten) Aufwand und Tests. Forscher konzentrieren sich nur selten auf Fragen der Benutzerfreundlichkeit und Forschungseinrichtungen fehlen die finanziellen Mittel und Arbeitskräfte, um benutzerfreundliche Software zu entwickeln. Auch wenn heutzutage viele gute Bibliotheken für die UI-Entwicklung existieren, erfordert dies immer noch viel Zeit und Mühe und teilweise andere Programmierkenntnisse, als sie bei wissenschaftlichen Mitarbeitern üblicherweise vorhanden sind.

Der Einsatz und die Kombination einiger leicht verständlicher Open-Source-Werkzeuge kann diesen Arbeitsaufwand erheblich verringern. Im Rahmen verschiedener betreuter Abschlussarbeiten wurde ein Konzept zur digitalen Integration einer am Institut entwickelten pseudokontinuierlichen Chromatographieanlage (engl. *Periodic Counter Current Chromatographie,* PCCC) ausgearbeitet. Dazu wird der SiLA2-Standard für die Kommunikation von Laborgeräten und die *Google-Blockly*-Bibliothek zur einfachen *Point-and-Click*-basierten Methodenerstellung und Parametrisierung genutzt.

Es sollte gezeigt werden, dass auch dann, wenn eine zukünftige Integration während der Designzeit eines Gerätes nicht geplant war, dieses Ziel durch den Einsatz frei verfügbarer Werkzeuge ohne größere Änderungen der zugrundeliegenden Techniken erreicht werden kann. Dieser Ansatz wird möglicherweise nicht als einfaches Design angesehen, ist aber von Nicht-Experten in der Software-Entwicklung mit geringem Aufwand an Zeit und Geld umsetzbar. Da alle Mechanismen gegenüber dem Zustand, in dem sie während der Designzeit der Anlage waren, unverändert bleiben, sind alle zuvor verwendeten Werkzeuge und die Infrastruktur weiterhin nutzbar.

3.7.1 AUSGANGSSITUATION

Das PCCC-System wurde bereits getestet und charakterisiert [9–12]. Es wird jedoch noch nicht von anderen Wissenschaftlern des Instituts eingesetzt, obwohl viele großes Interesse bekundeten. Dies liegt vor allem an der Notwendigkeit, Methoden-Skripte in Python zu schreiben und das Gerät durch eine ferngesteuerte textbasierte *Secureshell* (SSH)-Anmeldung auf dem in das Gerät integrierten *Single Board Computer* (SBC) zu steuern. Dieser Computer wird in diesem Text als „PCCC-Kontrollsystem“ (PCS) bezeichnet.

Das Gerät verwendet vier Membranadsorber zur Durchführung der chromatographischen Proteintrennung. Es enthält eine Vielzahl von Ventilen, die auf der Grundlage von Signalen der Photospektrometer, die direkt hinter den Adsorbern angeordnet sind, geschaltet werden müssen. Auch Pumpen für Feed-, Elutions- und Waschlösungen müssen gesteuert werden. Durch den parallelen Einsatz dieser Komponenten kann eine Pseudokontinuität in der Reinigung erreicht werden [11].

Die Pumpen werden jeweils über eine serielle Schnittstelle angesteuert. Zur Steuerung der Wege wird ein Feldbuskoppler durch Setzen von Bits in seinem Modbus-Server zum Öffnen/Schließen von

Ventilen angewiesen. Die Spektrometer werden von einem separaten SBC über eine seriell-über-USB-Verbindung gesteuert. Auf diesem SBC läuft auch ein Modbus-Server, der die Konfiguration der Spektrometer ermöglicht und auch deren Messwerte darstellt. Eine separate kleine Anwendung auf einem der PCs im Labor musste verwendet werden, um die Photometer vor ihrem Einsatz zu konfigurieren. Während eines Chromatographielaufs wurden die Photometerwerte mit der Software *Delphin ProfiSignal* visualisiert (es kann jedoch jede andere Visualisierungslösung verwendet werden, die so konfiguriert werden kann, dass sie die Rohdaten von einem Modbus-Server liest).

Die gesamte Logik zur Steuerung der Pumpen und Ventile ist in Python-Klassen verpackt. Ein „PCCC-Hauptprogramm“ (PMP) instanziiert diese Klassen für jede Komponente im Gerät. Um eine Chromatographie-Methode zu konfigurieren, muss ein Python-Skript geschrieben werden, das diese Instanzen verwendet. Hier müssen alle notwendigen Methoden zum Öffnen/Schließen von Ventilen entsprechend der aktuellen Messwerte der Photometer in der richtigen Reihenfolge aufgerufen werden, um sinnvolle "Pfade" durch das Gerät zu bilden. Dies führt zu langen, sich wiederholenden Methodenskripten voller redundantem Quelltext. Diese Skripte sind schwer zu schreiben und zu warten, müssen aber regelmäßig geändert werden, um die Prozessparameter an neue Aufgaben anzupassen.

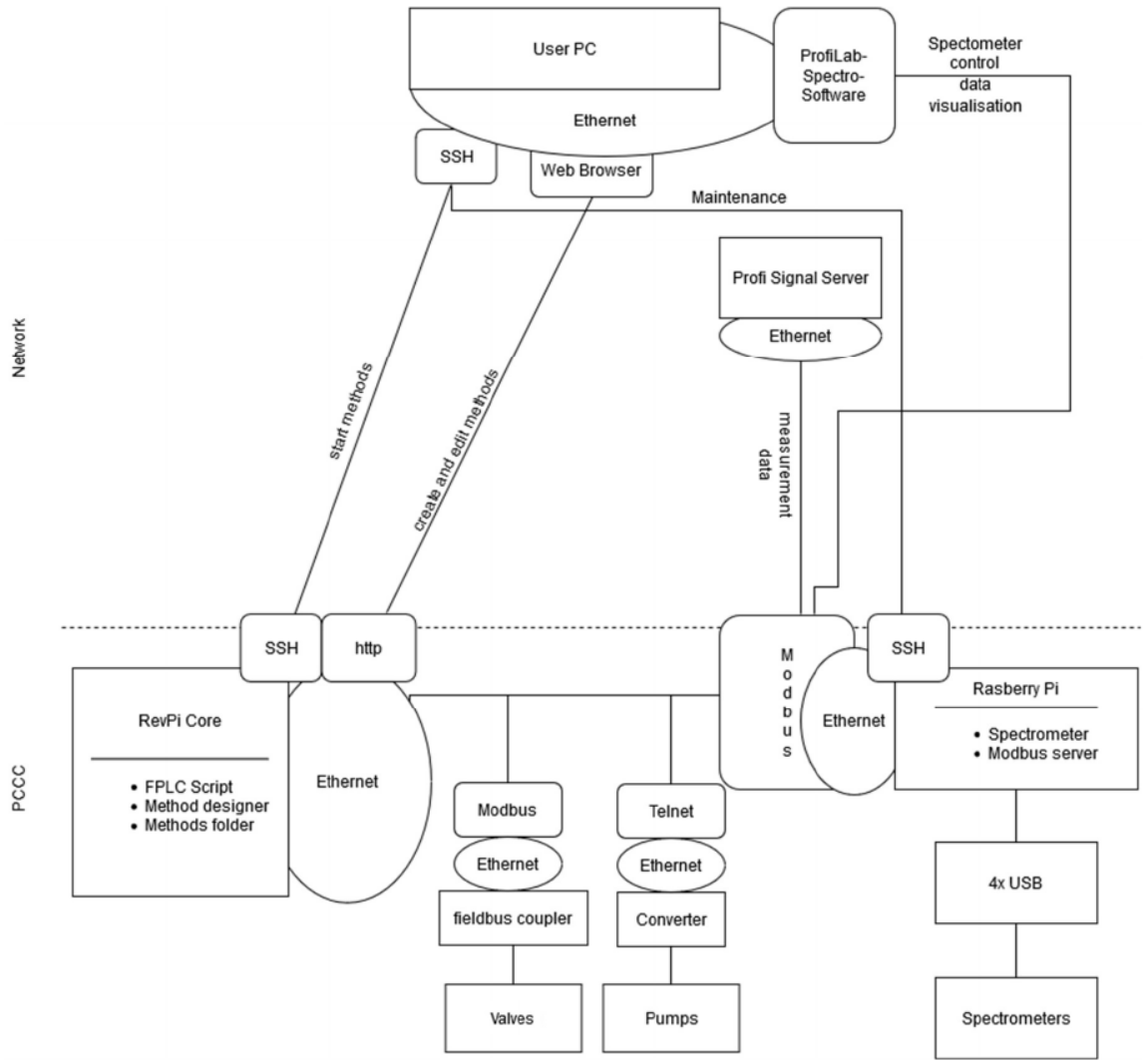


ABBILDUNG 27: KOMPONENTEN DES PCCC-SYSTEMS IM AUSGANGSZUSTAND. (DIE GRAFIK IST VON FRAU MICHELLE ANGELICA DJUARI IM RAHMEN IHRER BACHELORARBEIT ANGEFERTIGT WORDEN.)

Außerdem müssen die Methodenskripte in einem lokalen Ordner des Dateisystems des PCS abgelegt werden, wo das PMP sie finden kann. Um eine Methode auszuführen, muss sich der Benutzer über SSH mit dem *headless* (d. h. ohne direkte Peripheriegeräte) betriebenen PCS verbinden und das PMP zusammen mit einem Methodenskript ausführen.

In Abbildung 27 ist der Aufbau des PCCC-Systems dargestellt. Das System wird von zwei SBCs betrieben, ein *Raspberry Pi* (Steuerung der Spektrometer) und ein *RevPi Core* (PCS). Die Kommunikation aller Komponenten der Anlage erfolgt über Ethernet. Um die PCCC Anlage zu bedienen, werden vier verschiedene Programme verwendet. Der Raspberry Pi ist wegen der Verwendung von SD-Karten (*Secure Digital Memory Cards*) als Boot-Speichermedium nicht für einen kontinuierlichen Betrieb geeignet. Der *RevPi Core* ist außerdem mit einem ARM (*Advanced RISC (Reduced Instruction Set Computer) Machines*)-Prozessor älterer Bauart ausgestattet, auf dem gRPC (als Basis für SILA2) nicht lauffähig ist.

3.7.2 Hardware-Update und Digitalisierungskonzept

Um die Systemanforderungen zur Integration mit SiLA2 zu erfüllen, soll zunächst ein Upgrade der in der PCCC Anlage verwendeten Hardwarekomponenten durchgeführt werden, bei dem alle Software-Komponenten auf einem performanten SBC zusammengefasst werden. Eine System-Architektur soll ausgearbeitet werden, die auch eine Vereinfachung der Anlage vornimmt. Es soll auch eine Softwarekomponente zur Datenerhebung nach den FAIR-Datenprinzipien integriert werden.

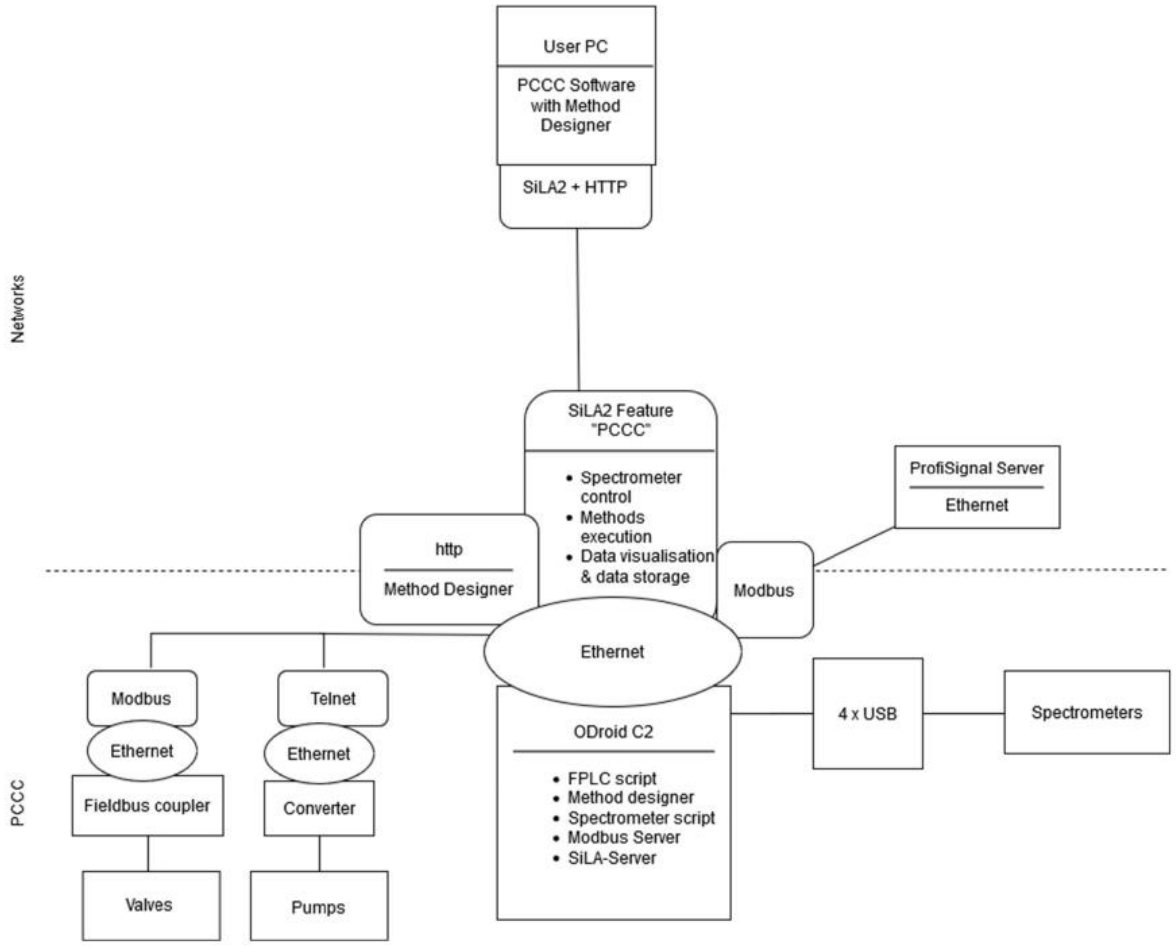


Abbildung 28: Komponenten des PCCC-Systems nach Hardware-Update und Vereinfachung. (Die Grafik ist von Frau Michelle Angelica Djuari im Rahmen ihrer Bachelorarbeit angefertigt worden.)

Das PCCC-System soll um eine benutzerfreundliche grafische Oberfläche erweitert werden. Als schnell umsetzbare Lösung wird ein webbasierter Methodendesigner unter Verwendung von *Google Blockly* entwickelt, der eine intuitiven *Point-and-Click*-Bedienung ermöglicht. Der Benutzer soll in der Lage sein, jeden Desktop-Computer mit einem Web-Browser zu benutzen, um schnell Chromatographie-Methoden zu erstellen und zu parametrisieren. Es soll nicht erforderlich sein, das Schreiben von Python-Code zu erlernen oder sich mit der Übertragung der Methoden-Skripte auf das Gerät zu beschäftigen.

Weiterhin wurde die Bedienung der Programme, die die PCCC steuern, in einen SiLA2 Server verpackt, der auf dem PCS läuft und die Ausführung der zuvor entworfenen Methoden vom Web-Methodendesigner anbietet. Da das SiLA2-Protokoll verwendet wird, ist die Anbindung an diesen PCCC-Server von überall im Netzwerk über einen entsprechenden SiLA2-Client problemlos möglich. Zu diesem Zweck soll eine PCCC-Steuersoftware entwickelt werden.

Abbildung 28 zeigt das PCCC-System nach Umsetzung des Digitalisierungskonzepts. Die Inhalte der zwei SBCs werden auf einen neuen SBC übertragen. Dazu wird ein *ODroid C2* verwendet, da dieser mit einem aktuellen ARM-Prozessor der 8. Generation ausgestattet ist und einen *embedded MultiMediaCard Connector* (eMMC) aufweist. Ein eMMC hat eine schnellere Leserate und ist robuster als eine SD-Karte. Die Steuerung der Anlage erfolgt über SiLA2. Die Steuerung und die Wartung der PCCC Anlage und das Aufrufen der Methoden erfolgen in einer gemeinsamen Software. Ein Methodendesigner unterstützt den Benutzer grafisch.

Um die während eines Methodenlaufs erzeugten Daten FAIR-konform zu speichern, soll eine Software entwickelt werden, die AnIML-Dateien erzeugt (ein sog. AnIML-Generator). Abbildung 29 zeigt den Ablauf einer Methode im Kontext der FAIR-konformen Datenaufnahme.

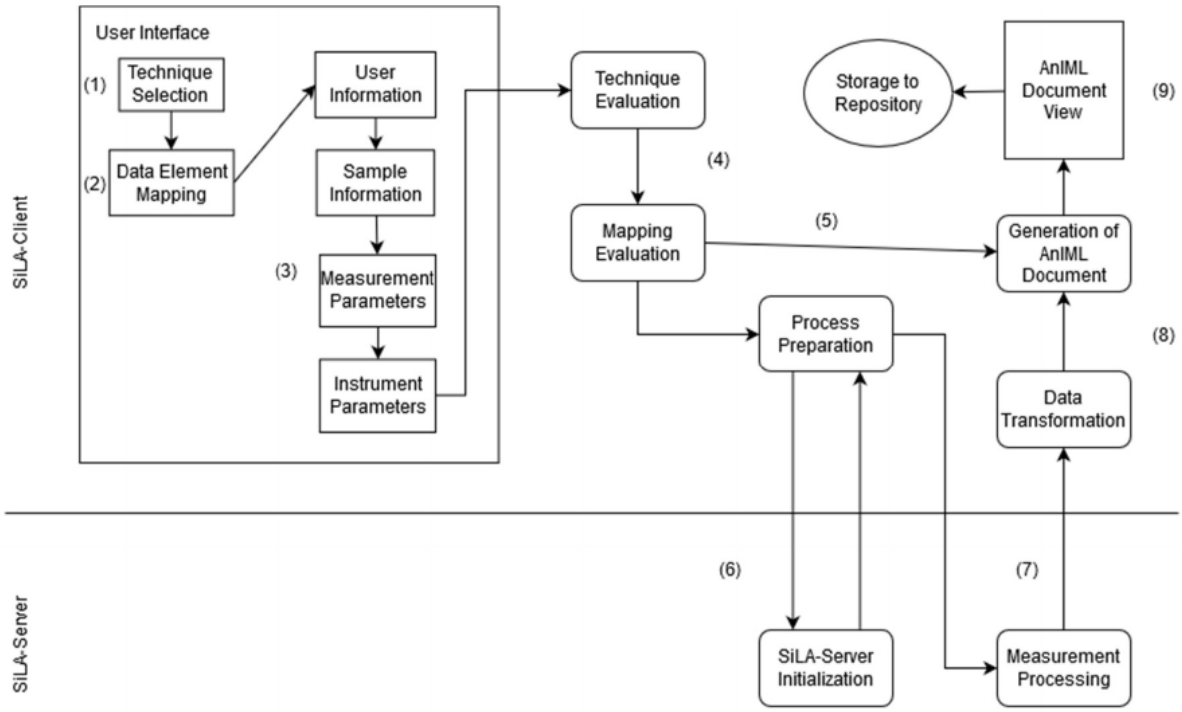


ABBILDUNG 29: ABLAUF EINER CHROMATOGRAPHIEMETHODE. (DIE GRAFIK IST VON FRAU MICHELLE ANGELICA DJUARI IM RAHMEN IHRER BACHELORARBEIT ANGEFERTIGT WORDEN.)

3.7.3 Digitale Anbindung

Zur Umsetzung der digitalen Anbindung werden SiLA2 [70–72] und *Google Blockly* [6] verwendet. Blockly ist eine Open-Source-JavaScript (js)-Bibliothek von *Google Developers*, die grafische *Point-and-Click*-Programmiereditoren für webbasierte Anwendungen erstellt. Sie wurde ursprünglich entwickelt, um das Unterrichten von Programmiersprachen zu erleichtern, indem sie eine visuelle Schnittstelle zur Kontrolle von Strukturen wie Schleifen und If-Bedingungen bietet. Blockly kann komplexe Programme in jeder Sprache aus vordefinierten Elementen (so genannten "Blöcken") erstellen, die in einem visuellen Editor angeordnet werden können. Sogenannte Code-Generatoren legen fest, wie jeder Block in Quellcode übersetzt wird.

Blockly wird als komplette Client-seitige js-Bibliothek geliefert, die leicht in jede Webseite integriert werden kann. Die dem Benutzer zur Verfügung gestellten Blöcke können frei spezifiziert werden. Es ist nicht notwendig, Blöcke zu verwenden, die sich direkt in einfache Ausdrücke im Quellcode übersetzen lassen. Es ist auch möglich, komplexe, aber redundante Logik hinter einem einzigen Block zu verstecken und dem Benutzer lediglich die Möglichkeit zu bieten, den Arbeitsablauf zu parametrisieren. Da Blöcke und die entsprechenden Code-Generatoren frei auf spezielle Bedürfnisse zugeschnitten werden können, lässt sich Blockly leicht anpassen, um Steuerskripte für kundenspezifische komplexe Geräte zu erstellen und zu modifizieren.

Der Blockly-Methodendesigner wurde erzeugt, indem eine Webseite erstellt wurde, die den Blockly-Editor auf der rechten Seite des Bildschirms und ein Menü zum Laden oder Neu-Erstellen von Methodendateien auf der linken Seite präsentiert. Dieses Menü verwendet kleine PHP (Rekursives Akronym: PHP Hypertext Preprocessor)-Skripte, um den Inhalt des Methodenordners auf dem PCS zu lesen und Methoden zurück auf dem PCS zu speichern. Diese Skripte werden von js-Funktionen in der HTML-Datei des Methodendesigners verwendet. Da Blockly nicht in der Lage ist, den Quellcode zurück in Blöcke zu übersetzen, müssen für jede Methode zwei Dateien gespeichert werden. Eine davon ist das Python-Methoden-Skript, das vom PMP ausgeführt werden kann. Die andere ist eine XML-Datei, die durch eine Blockly-Methode erzeugt wird und die Blockstruktur speichert.

Der Blockly-Editor wird mit Hilfe der *Inject*-Methode der Blockly-Bibliothek eingefügt. Zur Festlegung der Blöcke, die dem Benutzer präsentiert werden, wird eine so genannte *Toolbox* im XML-Format erstellt, die nur aus speziell für die PCCC-Anwendung entwickelten Blöcken besteht. Diese

```
{
  "type": "wait_seconds",
  "message0": "Wait %1 seconds",
  "args0": [
    {
      "type": "field_number",
      "name": "seconds",
      "value": 0,
      "min": 0,
      "precision": 1
    }
  ],
  "previousStatement": null,
  "nextStatement": null,
  "colour": 230,
  "tooltip": "System wait a specific amount of time",
  "helpUrl": "www.help-url.de"
}
```

Abbildung 30: Beispiel einer JSON-Definition für einen frei definierten Blockly-Block. (Die Grafik ist von Frau Laura Niemeyer im Rahmen ihrer Bachelorarbeit angefertigt worden.)

```
Blockly.Python['wait_seconds'] = function(block) {
      var number_seconds = block.getFieldValue('seconds');
      var code = 'pause(' + number_seconds + ')' + '\n';
      return code;
      };
```

ABBILDUNG 31: BEISPIEL EINER JS-GENERATORFUNKTION FÜR EINEN FREI DEFINIERTEN BLOCKLY-BLOCK. (DIE GRAFIK IST VON FRAU LAURA NIEMEYER IM RAHMEN IHRER BACHELORARBEIT ANGEFERTIGT WORDEN.)

speziellen Blöcke müssen als JSON-Elemente definiert werden (siehe Abbildung 30), die das Aussehen der Blöcke beschreiben. Außerdem muss für jeden Block eine js-Funktion geschrieben werden, die eine Zeichenkette mit dem von diesem Block generierten Python-Quellcode zurückgibt (Abbildung 31).

Um einen einfachen Zugriff auf den Blockly-Methodendesigner zu ermöglichen, wird dieser von einem Webserver gehostet, der auf dem PCS läuft. Dies ermöglicht es dem Laborbenutzer, Methoden von jedem beliebigen PC im Labor aus zu erstellen und zu modifizieren und macht somit die Übertragung von Python-Skripten auf den PCS überflüssig. Als Web-Server-Lösung wird der weit verbreitete Open-Source-Server *Apache2* eingesetzt. Abbildung 32 zeigt einen Screenshot des Methoden-Designers.

Für die Implementierung des SiLA-Servers wurde die C# sila_tecan-Implementierung [69] verwendet, da sie nur geringen Implementierungsaufwand (Schreiben einer Schnittstelle) erfordert und allen notwendigen Overhead zur Erfüllung der SiLA-Spezifikationen und zur automatischen Durchführung der Netzwerkkommunikation über gRPC erzeugt. Obwohl das Tecan SDK (*Software Development Kit*) auf dem .NET Framework basiert, kann es zur Implementierung eines Servers in dotnet-Core verwendet werden [60]. Dies ist nützlich, da auf dem PCS ein Linux-Betriebssystem läuft.

Ein *Interface*, das das Feature beschreibt, wurde geschrieben und mit verschiedenen Attributen annotiert, die die Funktionalität in einer SiLA-kompatiblen Weise beschreiben. Anschließend wurde das *Interface* durch eine Klasse implementiert, die sie mit dem gesamten für den Betrieb der PCCC notwendigen Code füllt. Dies bedeutet, dass die Aufrufe an das PMP abgewickelt und die Messwerte über Modbus abgerufen werden. Schließlich wurde ein ausführbares Programm geschrieben,

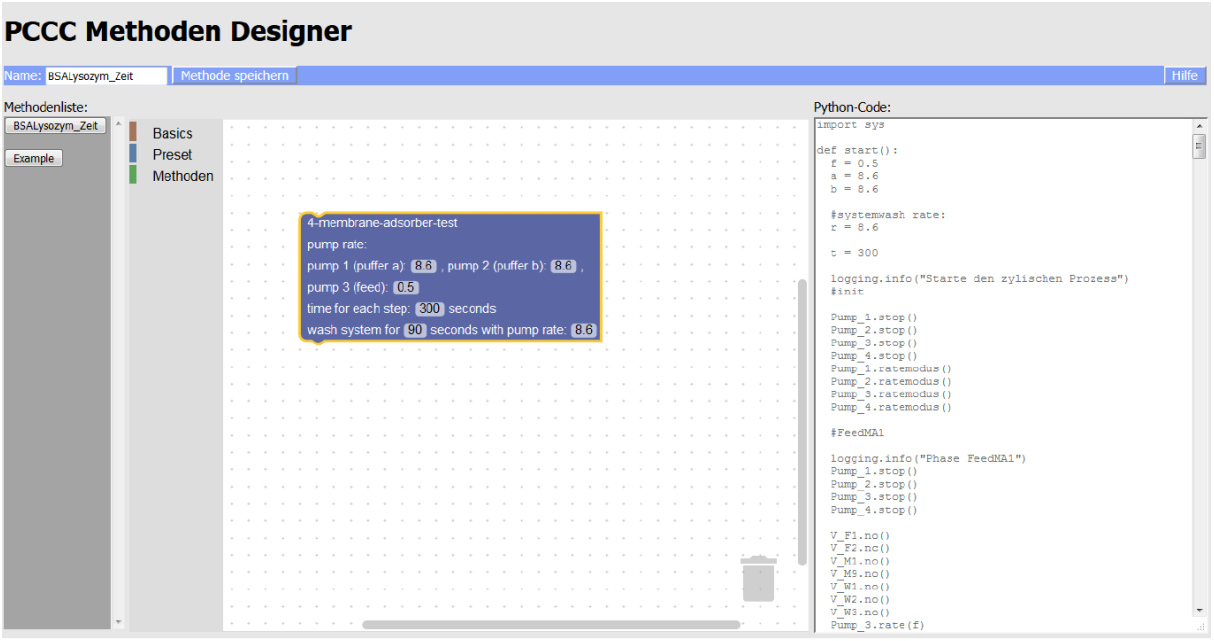


ABBILDUNG 32: SCREENSHOT DES PCCC-METHODENDESIGNERS MIT EINEM EINFACHEN BEISPIELABLAUF AUS EINEM EINZELNEN KOMPLEXEN BLOCK. (DIE GRAFIK IST VON FRAU LAURA NIEMEYER IM RAHMEN IHRER BACHELORARBEIT ANGEFERTIGT WORDEN.)

das den SiLA-Server startet. Dazu wurde der Code-Generator aus dem Tecan SDK verwendet, um die SiLA2-konforme Feature-Definition-XML-Datei und die notwendigen *Wrapper-Funktionen* zur Verwendung von SiLA2-kompatiblen Datentypen automatisch zu generieren.

Außerdem wurde eine ATDD für PCCC-Abläufe konzeptioniert und ein AnIML-Generator in der Sprache C# entwickelt, der AniML-konforme XLM-Dateien aus PCCC-Messergebnissen erzeugt. Dieser wird als Komponente der PCCC-Steuersoftware verwendet, um die Umsetzung der FAIR-Richtlinien optimal vorzubereiten.

Auf Basis der beschriebenen Komponenten befindet sich ein PCCC-Steuerprogramm zurzeit in der Entwicklung, dass die Bedienung der Anlage von jedem PC im Netzwerk aus ermöglicht und alle Funktionen in einer Oberfläche vereint. Diese Software wird als Web-Anwendung entwickelt, die ebenfalls vom Web-Server im PCS gehostet wird, so dass auf den Nutzer-PCs keine weitere Software benötigt wird.

3.7.4 Verwendung

Für den Betrieb des voll digital integrierten PCCC-Systems benötigt der Anwender lediglich einen PC mit Webbrowser. Zunächst kann eine neue Methode durch Zugriff auf den *Point-and-Click*-Methodendesigner erstellt werden. Diese Methode kann mit Hilfe der Web-basierten Steuersoftware direkt vom selben PC aus gestartet werden. Die Visualisierung der Spektrometermesswerte geschieht direkt in dieser Anwendung, es kann aber auch jede beliebige andere Datenvisualisierungslösung verwendet werden. Die Archivierung der Ergebnisdaten erfolgt ebenfalls automatisiert und standardisiert FAIR-konform mit Hilfe des AnIML-Generators.

3.8 NUTZERINTERAKTIONSSTUDIEN IM DIGITALISIERTEN LABOR

Um die Nutzbarkeit des digitalisierten Labors und die Akzeptanz der Nutzer für verschiedene Digitalisierungslösungen zu untersuchen, wurden Konzepte für Nutzerinteraktionsstudien erarbeitet und umgesetzt. Der Fokus lag dabei vor allem auf der Kommunikation des digitalen Systems mit dem Nutzer und damit auf der Untersuchung verschiedener UI-Geräte bei ihrem Einsatz in der digitalisierten Laborumgebung.

Dazu wurden Beispiel-Arbeitsabläufe im digitalisierten Labor durch Teilnehmer durchgeführt, die bisher keine Arbeitserfahrung mit dem digitalen Laborsystem hatten. Die Ergebnisse der Arbeit sowie die Reaktionen der Benutzer wurden dabei erfasst und untersucht.

3.8.1 KONZEPT

Bei den Nutzerinteraktionsstudien sollen die Benutzer im ersten Teil der Studie bestimmte SOPs im digitalisierten Labor ausführen, und im zweiten Teil eine Umfrage zu ihren Erfahrungen beim Arbeiten ausfüllen. Nachdem erst der inhaltliche Ablauf eines Arbeitsvorgangs festgelegt und entsprechend als Ablaufskript ausgearbeitet wird, kann ein Fragebogen zur Einschätzung für die Benutzer konzipiert werden. Um die Fragenbereiche zu gliedern, sollten Themenkonzepte für Mensch-Computer-Interaktionen, wie beispielsweise das Schema nach Ken Eason (1991), betrachtet werden (s. Abbildung 33) [22].

Dabei steht auf der ersten Stufe die direkte Interaktion zwischen Mensch und Computer. Die nächste Stufe umfasst die zu erfüllende Aufgabe und die technische Umgebung, u. a. den genutzten Computer, die Art von Bildanzeige (Monitor, Tablet, Datenbrille, etc.), sonstige Features (z. B. Sprachsteuerung) und – im Falle des digitalisierten Labors – auch das verfügbare Laborequipment. Auf der dritten Stufe befindet sich die Interaktion aus organisatorischen Gegebenheiten, dem technischen System und der sozialen Struktur. Auf dieser Ebene wird betrachtet, wie das technische System das Leben des Menschen (beispielsweise in Hinblick auf Arbeit oder Sozialleben) verändert [22, 84]. Eine detaillierte Auflistung der beeinflussenden Faktoren findet sich bei Jennifer Preece *et al.* (1994) [62]. Abbildung 34 zeigt eine grafisch aufbereitete Übersicht dieser Faktoren, sortiert nach ihrem Einflussbereich [84].

Bei der Analyse von Mensch-Computer-Interaktionen spielen nicht nur Faktoren bezüglich des Arbeitsauftrags und der physischen und digitalen Arbeitsumgebung eine Rolle, sondern auch der Nutzer selbst. Dieser besitzt neben seiner Persönlichkeit und Erfahrung auch noch individuelle Maße an Motivation [62, 84]. Die Interaktion zwischen Mensch und technischem System ist dabei komplex und abhängig vom Kontext, in dem das System eingesetzt wird [84]. Diese Einflussfaktoren sind in Abbildung 35 dargestellt.

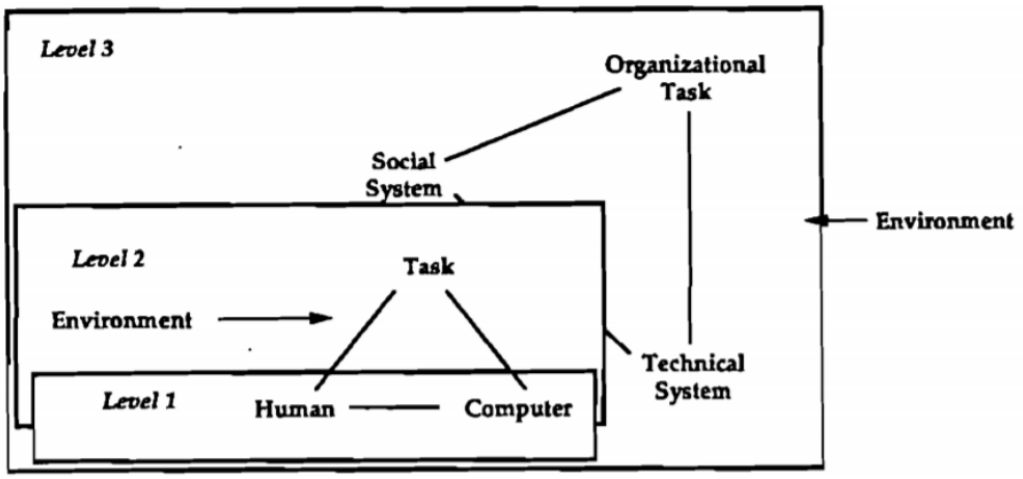


ABBILDUNG 33: MODELL DER MENSCH-COMPUTER-INTERAKTION NACH EASON. (BILDQUELLE: [22])

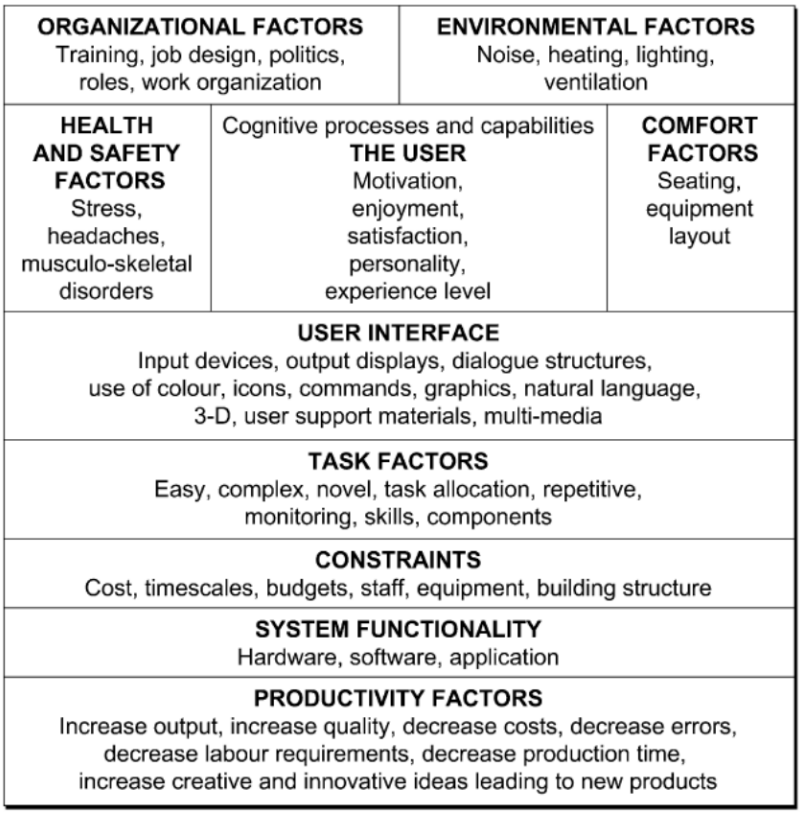


ABBILDUNG 34: EINFLUSSFAKTOREN AUF DAS EMPFINDEN VON TECHNISCHEN SYSTEMEN DURCH DEN BENUTZER . (BILDQUELLE: [84])

Für den Entwurf der Fragebögen wurde auf das online-Werkzeug *Google Forms* zurückgegriffen. Dieses ist leicht zu bedienen und bietet gleichzeitig flexible Möglichkeiten der Datenauswertung, indem die Ergebnisdaten aller durchgeführten Befragungen automatisiert zusammengefasst werden können. Neben diesen Befragungsergebnissen, werden auch die Zeiten, die die Probanden für die Durchführung der Arbeitsabläufe benötigen, erfasst. Abbildung 36 zeigt beispielhaft die Nutzeroberfläche beim Erstellen eines Fragenbogens.

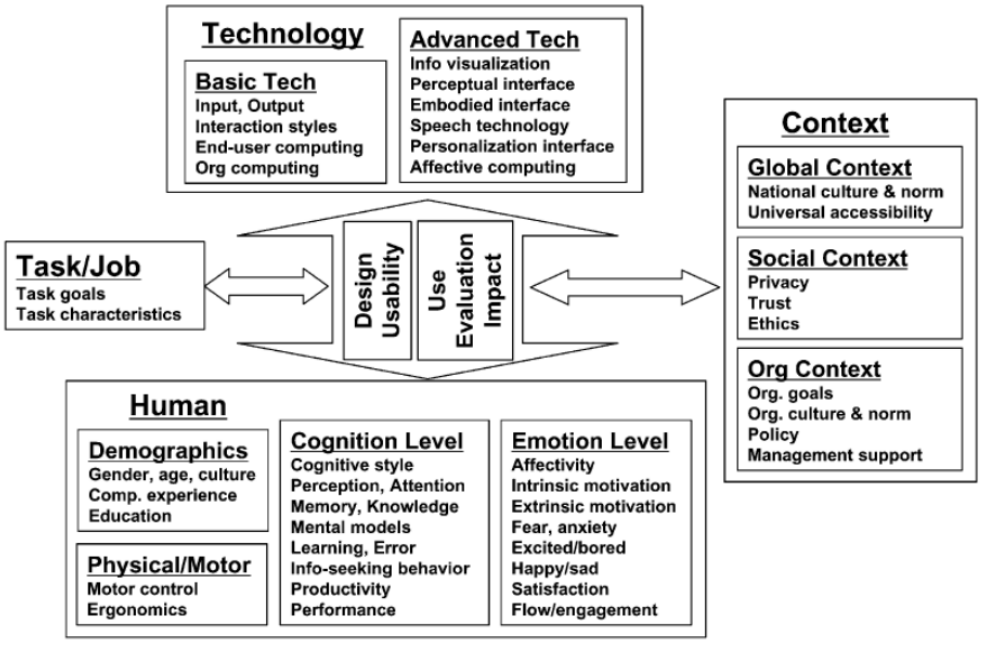


ABBILDUNG 35: ZUSAMMENSPIEL DES TECHNISCHEN SYSTEMS MIT DEM MENSCHLICHEN BENUTZER UNTER EINFLUSSNAHME DER RAHMENBEDINGUNGEN. (BILDQUELLE: [84])

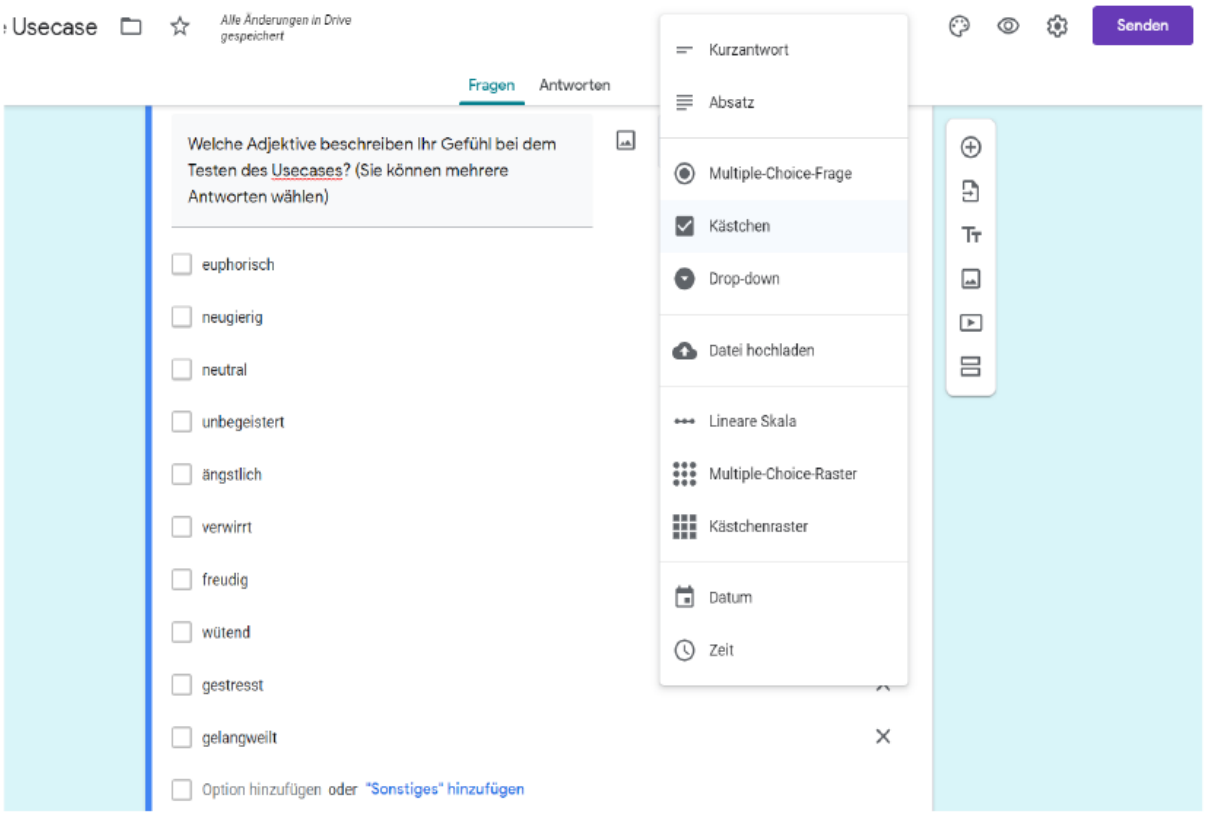


Abbildung 36: Screenshot der Eingabemaske von Google Forms zum Erstellen eines Fragebogens. (Die Grafik ist von Frau Karolina Sikora im Rahmen ihrer Bachelorarbeit angefertigt worden.)

Es wurde auf Basis der oben dargelegten Rahmenbedingungen ein Fragenkatalog ausgearbeitet, der die Vor- und Nachteile des digitalisierten Labors möglichst vollständig erfassen soll:

Abschnitt 1: Fragen zur Person und Laufbahn

- Wie alt sind Sie?
- Mit welchem Geschlecht identifizieren Sie sich?
- Sind, bzw. waren Sie berufstätig?
- Falls Sie berufstätig sind/waren: Haben Sie in Ihrer Laufbahn Arbeitserfahrungen im Laborbereich gemacht?
- Wenn ja, geben Sie bitte die Menge an Jahren als Zahl an.
- Falls Sie Schüler*in oder Student*in sind: Sind Sie naturwissenschaftlich interessiert?
- Falls Sie Student*in sind: Ist Ihr Studiengang naturwissenschaftlich geprägt?

Abschnitt 2: Technikerfahrungen

- Wie würden Sie Ihre Fähigkeiten im Umgang mit Technik einschätzen?
- Nutzen Sie Smart Assistants wie Amazons Alexa, Apples Siri oder Microsofts Cortana, den Google Assistant etc. aktiv im privaten Gebrauch?
- Wie nützlich finden Sie Smart Assistants für den privaten Gebrauch?
- Bitte geben Sie an, warum Sie Smart Assitants nützlich/unnütz finden.
- Wie viel Technik- und Digitalisierungsbezug erfahren Sie in Ihrem Arbeits-/Studiums-/Schulalltag?
- Erläutern Sie kurz, wie Sie in Ihrem Arbeits-/Studiums-/Schulalltag Kontakt mit Technik und Digitalisierung erfahren.

- Finden Sie, dass Ihr Arbeits-/Studiums-/Schulalltag von Technik und Digitalisierung profitiert/profitieren könnte?
- Erläutern Sie kurz, warum Technik und Digitalisierung (un-)vorteilhaft für Ihren Arbeits-/Studiums-/Schulalltag ist bzw. sein könnte.

ABSCHNITT 3: FRAGEN ZUR INHALTLICHEN KOMPLEXITÄT DES EXPERIMENTS

- Wie kompliziert fanden Sie den inhaltlichen Ablauf des Experiments?
- Haben Sie schon in Vergangenheit ähnliches Experiment durchgeführt?
- Falls Sie in Vergangenheit ein ähnliches Experiment durchgeführt haben: Wie oft haben Sie dies (geschätzt) getan?

ABSCHNITT 4: FRAGEN ZUM ABLAUF UND LAYOUT

- Mit welchem UI-Gerät haben Sie das Experiment durchgeführt?
- Wie schätzen Sie ihre Motivation bzw. ihr Interesse vor dem Start ein?
- Welche Adjektive beschreiben Ihr Gefühl bei dem Testen des Experiments? (ängstlich, verwirrt, freudig, wütend, gestresst)
- Wie schätzen Sie ihre Motivation bzw. ihr Interesse während des Ablaufs ein?
- Geben Sie Ihre Meinung zu folgenden Punkten an:
    - Das Experiment war leicht
    - Ohne die digitale Begleitung wäre das Experiment schwierig durchzuführen
    - Die Durchführung des Experiments lief durch die digitale Begleitung schnell
    - Die Layouts waren anschaulich
    - Die Layouts waren verständlich
    - Die Layouts haben mich gut durch das Experiment begleitet
    - Die Layouts hatten ein schönes Design
    - Obwohl ich keine Erfahrungen im Laborbereich habe, konnte ich dieses Experiment durch die digitalisierten Abläufe problemlos durchführen
    - Ich habe Laborerfahrung und fand die Digitalisierung des Arbeitsablaufs sehr sinnvoll
- Fanden Sie dieses digitalisierte Experiment insgesamt sinnvoll?

3.8.2 ERSTE ERGEBNISSE

In einer ersten praktischen Anwendung des dargelegten Konzepts wurde durch einige Studenten eine Nutzerinteraktionsstudie mit drei verschiedenen Arbeitsabläufen durchgeführt. Es wurden Protokolle zum Ansetzen eines Kultivierungsmediums, zum Beproben eines Bioreaktors und zur Durchführung eines enzymatischen Amylase-Verdaus von Stärke entwickelt. Alle drei SOPs wurden für die Durchführung mit SmartGlasses, einem Tablet und ohne digitale Unterstützung mit papierbasierter SOP-Anleitung und Dokumentation vorbereitet.

Kein Proband hat einen Ablauf mehrmals oder verschiedene Abläufe mit der gleichen Nutzerinteraktionsmethode durchgeführt. Der Strichprobenumfang in dieser ersten Studie betrug 41 Personen. Abbildung 37 zeigt eine Übersieht der allgemeinen Informationen zu den Probanden.

Abbildung 38 zeigen die Ergebnisse der Nutzerbefragung nach Durchführung der SOPs jeweils aufgeteilt auf die verwendeten UI-Devices. Die Befragung wurde mit leicht von den oben dargestellten Fragen abweichenden Inhalten durchgeführt, da der Fragekatalog zum Zeitpunkt des Starts der Studie noch nicht zur Verfügung stand.

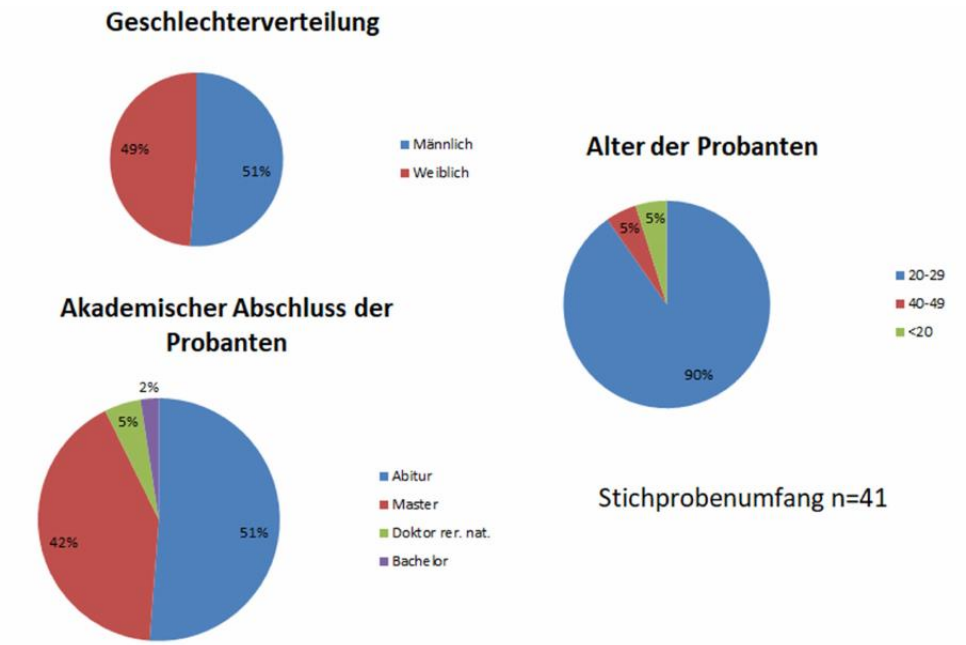


ABBILDUNG 37: ALLGEMEINE INFORMATIONEN ÜBER DIE PROBANDEN DER NUTZERINTERAKTIONSSTUDIE.

Bei der Nutzerzufriedenheit ist zu erkennen, dass die Verwendung der digitalen Unterstützung im Allgemeinen als nützlich empfunden wird, die SmartGlasses bei der Darstellung und dem ansprechenden Aussehen der Anweisungen jedoch deutlich schlechter abschneiden, als das Tablet. Dies kann darauf zurückgeführt werden, dass ein Großteil der Probanden keine Erfahrungen mit dieser Technologie hatte und die Verwendung der SmartGlasses eine gewisse Einarbeitungszeit erfordern, bis damit sinnvoll gearbeitet werden kann und sie als angenehm empfunden werden [14]. Mindestens eine Form der digitalen Unterstützung wurde in allen Kategorien im Durchschnitt besser bewertet als die papierbasierte Durchführung.

Abbildung 39 zeigt die durchschnittlichen Durchführungszeiten der Abläufe. Bei dem sehr dokumentationsintensiven Ablauf zur Medienherstellung (Einwaagen und Chargennummern der verwendeten Chemikalien mussten dokumentiert werden) bringt die digitale Unterstützung einen Zeitgewinn gegenüber der papierbasierten Durchführung. Da die Dokumentation von vielen Probanden nicht in dem Umfang durchgeführt wurde, wie sie im Sinne einer guten Nachvollziehbarkeit wün-

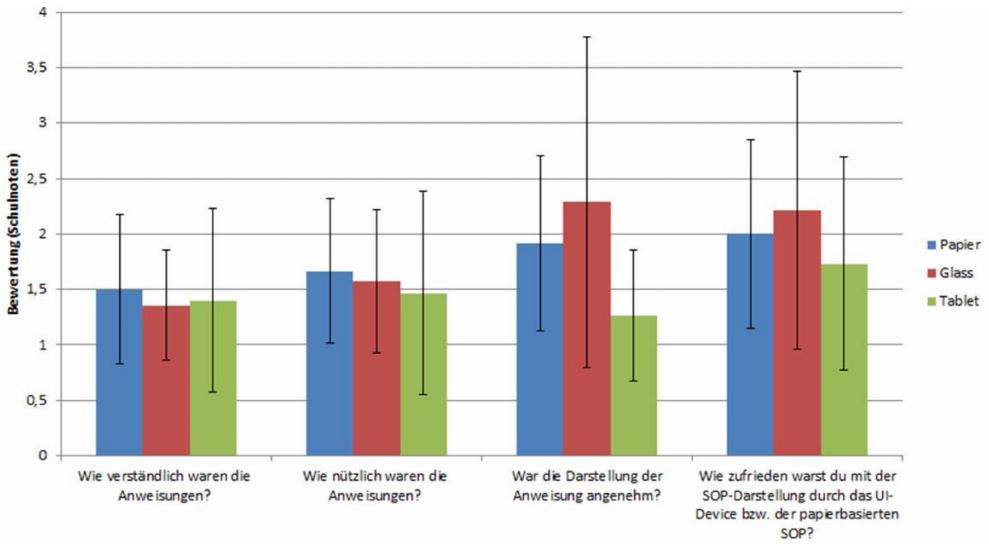


ABBILDUNG 38: DARSTELLUNG DER BEFRAGUNGSERGEBNISSE ZUR NUTZERZUFRIEDENHEIT. BEWERTUNG IN SCHULNOTEN: JE KLEINER DIE SÄULE, DESTO HÖHER DIE BEWERTUNG.

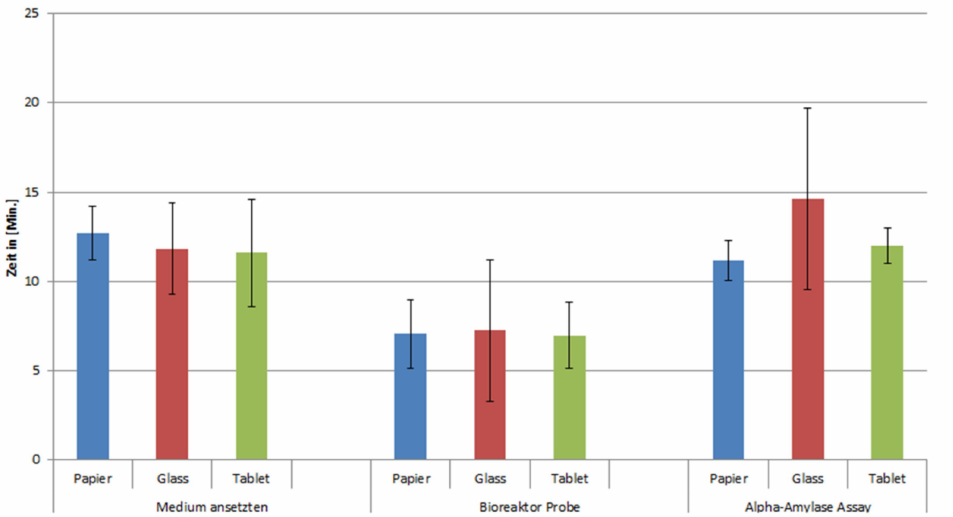


ABBILDUNG 39: DARSTELLUNG DER DURCHSCHNITTLICHEN ZEITEN, DIE DIE PROBANDEN FÜR DAS DURCHFÜHREN DER ARBEITEN MIT DEN VERSCHIEDENEN UI-GERÄTEN BENÖTIGTEN.

schenswert gewesen wäre, bedeutet die automatische, digitale Dokumentation hier zusätzlich einen Qualitätsgewinn. Die Beprobung des Bioreaktors erforderte wenig Interaktion mit dem digitalen Laborsystem und wenig Dokumentation; hier liegen alle Varianten in etwa gleich auf. Die SmartGlasses verlangsamen den Ablauf geringfügig (vermutlich aus den oben genannten Gründen). Bei dem enzymatischen Verdau, der viele Warte- und Pipettierzeiten und wenig digitale Interaktion beinhaltete, ist die Durchführung ohne digitale Unterstützung am schnellsten, da die Vorteile des digitalisierten Labors hier nicht ins Gewicht fallen konnten. Interessant ist, dass die Durchführung mit SmartGlasses hier deutlich langsamer ist, als mit Hilfe des Tablets. Da die zeitlichen Effekte durch die Gewöhnung an dieses Interaktionsmedium sich bei den anderen Abläufen nicht in diesem Umfang zeigen, sollte dieser Unterschied weiter untersucht werden.

Die oben erläuterten Ergebnisse legen nahe, dass das digitalisierte Labor einige vielversprechende Ansätze zur Vereinfachung und Verbesserung der Arbeitsabläufe im biotechnologischen Labor bietet. Auf dieser Grundlage sollten in Zukunft weitere Nutzerinteraktionsstudien durchgeführt werden können, die die Basis für einen kontinuierlichen Weiterentwicklungsprozess des entwickelten digitalen Laborsystems bilden.

4 ZUSAMMENFASSUNG UND AUSBLICK

Im Rahmen der vorliegenden Arbeit wurde erfolgreich eine Systemarchitektur konzeptioniert, implementiert und validiert, die die digitale Transformation existierender biotechnologischer Labore erlaubt. Das Konzept ist übertragbar auf andere Laboratorien mit unterschiedlichen Automatisierungsgraden der durchgeführten Arbeiten.

Laborgeräte und andere Ressourcen sowie Services werden einheitlich über den SiLA2 Kommunikationsstandard angebunden. Für Geräte, die SiLA2 nicht von Haus aus unterstützen, werden Gateways eingesetzt. Um die Einbindung von Bestandsgeräten ohne Ethernet-Schnittstelle in das Labornetzwerk zu erlauben, wurde ein Gateway-Modul auf Basis eines Embedded-Computers entwickelt. Die SiLA2 Server werden in einer Microservice-basierten Architektur strukturiert und können durch die entwickelten Methoden zum automatischen Verteilen und Steuern der Komponenten zentral verwaltet werden.

Ein zentraler Server stellt die Schnittstelle zum digitalisierten Labor dar. Er verbindet sich als SiLA2 Client mit allen im Labor vorhandenen Geräten und stellt deren Funktionen über eine zentrale REST-API zur Verfügung. Dabei werden alle Daten, die im Labor erzeugt werden, in einer Datenbank gespeichert und können jederzeit (auch ohne dass das erzeugende Gerät erreichbar sein muss) zugegriffen werden. Der Laborserver dient mit seiner REST-API dazu ebenfalls als zentrale Anlaufstelle und agiert als Datenbroker.

Im Rahmen dieser Dissertation wurde ebenfalls ein Prozessleitsystem entwickelt, dass mit der REST-API des Laborservers kommuniziert und Ablaufprotokolle im digitalisierten Labor ausführen kann. Der Fokus lag dabei auf der intuitiven Interaktion mit den Menschen im Labor. Dieses Prozessleitsystem erlaubt die Formulierung von Laborabläufen in einer stark abstrahierten Form, und ermöglicht so das Beschreiben von SOPs ohne Kenntnisse von den konkreten Mechanismen zur Gerätekommunikation. Durch die standardisierten Schnittstellen sind die Geräte im Labor einfach austauschbar ohne die SOP-Skripte anpassen zu müssen.

Die Kommunikation des Laborsystems mit dem Benutzer während der Prozessausführung findet über eine generische Nutzerinteraktionsschnittstelle statt. Die einzelnen UI-Geräte (wie SmartGlasses, Sprachsteuerung, Tablets, etc.) werden in der Protokollerstellung abstrahiert, sodass SOPs unabhängig von den verwendeten Nutzerinteraktionsmedien formuliert werden können.

Die Praxistauglichkeit der entwickelten Lösung konnte anhand von Nutzerinteraktionsstudien und der Adaption von Beispiel-Abläufen bereits gezeigt werden. In weiteren Arbeiten sollten nun Standardprozeduren in das digitalisierte Labor übertragen werden, um von den Vorteilen der digitalen

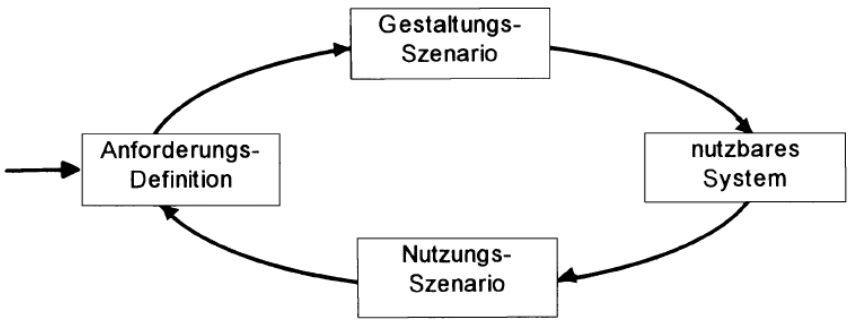


ABBILDUNG 40: DER *TASK-ARTIFACT-CYCLE* KANN ALS .MECHANISMUS ZUR KONTINUIERLICHEN VERBESSERUNG VON KOMPLEXEN SYSTEMEN – WIE DEM DIGITALISIERTEN LABOR – HIN ZU EINER OPTIMALEN ANWENDBARKEIT DIENEN. (BILDQUELLE: [44])

Unterstützung und der automatischen und strukturierten Dokumentation zu profitieren. Das Erzeugen von Ergebnisdaten, die den FAIR-Prinzipien folgen, konnte ebenfalls erfolgreich durchgeführt werden. Auf dieser Basis sollten in weiteren Arbeiten große Datenmengen zu bestimmten wissenschaftlichen Fragestellungen erzeugt werden können, die dann als Basis für Big-Data-Analysemethoden dienen können.

Der von John Carroll 1991 beschriebene *Task-Artifact-Cycle* (siehe dazu Abbildung 40) ist ein System, das dazu dient, technische Systeme kontinuierlich zu verbessern, indem das System in jeder Iteration gegen die gestellten Anforderungen geprüft und weiter optimiert wird [15]. Dieses Verfahren lässt sich auf Softwaresysteme übertragen. Dabei fließen in jedem Schritt Nutzererfahrungen in die Weiterentwicklung des Systems ein und bringen es so kontinuierlich näher an den Zustand, den der Benutzer als optimal empfindet [41, 44]. Dieses Vorgehen kann und sollte auch als Vorbild für die weitere Entwicklung des digitalisierten Labors dienen. Dazu sollten die begonnenen Nutzerinteraktionsstudien weitergeführt und verbessert werden. Die in Kapitel 3.8 (Seite 125) beschriebenen Arbeiten bieten hier eine Grundlage, auf der weiter aufgebaut werden kann.

Zum Senken der Einstiegshürde bei der Verwendung des digitalisierten Labors sollte eine grafische Anwendung zur Formulierung von SOPs für das Prozessleitsystem entwickelt werden. Aufgrund des hohen Abstraktionsniveaus der Ablaufskripte kann dies sehr einfach durch die Verwendung von Bibliotheken zur grafischen Programmierung (z. B. *Google Blockly*) erreicht werden.

Es wurde gezeigt, dass die digitale Transformation des biotechnologischen Labors zwar zeit- und arbeitsintensiv ist, aber auf Basis der heutigen technologischen Möglichkeiten gut durchführbar ist.

5 BIBLIOGRAPHIE

[1] Allotrope, Rethinking Scientific Data: *https://www.allotrope.org/*. Accessed: 2020-10-01.

[2] ASNeG - Automation Service of Next Generation, Open-source OPC UA solutions and client/server SDK in C++: *https://asneg.github.io/*. Accessed: 2020-10-01.

[3] Austerjost, J., Bargholz, M., Porr, M., Marquard, D., Lindner, P., Scheper, T., Beutel, S., Geier, D. and Becker, T. 2019. A flexible IT infrastructure for the integration of smartglasses into the brewing laboratory as a digital support for standard analysis workflows. *BrewingScience*. 72, 1–2 (2019), 1–9. DOI:https://doi.org/10.23763/BrSc18-20austerjost.

[4] Bär, H., Hochstrasser, R. and Papenfuß, B. 2012. SiLA: Basic standards for rapid integration in laboratory automation. *Journal of Laboratory Automation*. 17, 2 (2012), 86–95. DOI:https://doi.org/10.1177/2211068211424550.

[5] Bär, H. and Syré, U. 2011. Infoteam sila library simplifies device integration. *Journal of Laboratory Automation*. 16, 5 (2011), 371–376. DOI:https://doi.org/10.1016/j.jala.2011.05.003.

[6] Blockly, A JavaScript library for building visual programming editors: *https://developers.google.com/blockly*. Accessed: 2020-10-04.

[7] Blow, N. 2008. Lab automation: Tales along the road to automation. *Nature Methods*. 5, 1 (2008), 109–112. DOI:https://doi.org/10.1038/nmeth0108-109.

[8] Boeckhout, M., Zielhuis, G.A. and Bredenoord, A.L. 2018. The FAIR guiding principles for data stewardship: Fair enough? *European Journal of Human Genetics*. 26, 7 (2018), 931–936. DOI:https://doi.org/10.1038/s41431-018-0160-0.

[9] Brämer, C., Ekramzadeh, K., Lammers, F., Scheper, T. and Beutel, S. 2019. Optimization of continuous purification of recombinant patchoulol synthase from Escherichia coli with membrane adsorbers. *Biotechnology Progress*. 35, 4 (2019), 1–10. DOI:https://doi.org/10.1002/btpr.2812.

[10] Brämer, C., Lammers, F., Scheper, T. and Beutel, S. 2019. Development and testing of a 4-columns periodic counter-current chromatography system based on membrane adsorbers. *Separations*. 6, 4 (2019), 8–10. DOI:https://doi.org/10.3390/separations6040055.

[11] Brämer, C., Schreiber, S., Scheper, T. and Beutel, S. 2018. Continuous purification of Candida antarctica lipase B using 3-membrane adsorber periodic counter-current chromatography. *Engineering in Life Sciences*. 18, 7 (2018), 414–424. DOI:https://doi.org/10.1002/elsc.201700159.

[12] Brämer, C., Tünnermann, L., Salcedo, A.G., Reif, O.W., Solle, D., Scheper, T. and Beutel, S. 2019. Membrane adsorber for the fast purification of a monoclonal antibody using protein a chromatography. *Membranes*. 9, 12 (2019). DOI:https://doi.org/10.3390/membranes9120159.

[13] Brause, R. 2004. *Kompendium der Informationstechnologie*. Springer Berlin Heidelberg New York.

[14] Bundesanstalt für Arbeitsschutz und Arbeitsmedizin (BAuA) 2016. Head-Mounted Displays - Arbeitshilfen der Zukunft. (2016). DOI:https://doi.org/10.21934/baua:praxis20160809.

[15] Carroll, J.M., Kellogg, W.A. and Rosson, M.B. 1991. The Task-Artifact Cycle. *Designing Interaction: Psychology at the Human-Computer Interface*. Cambridge University Press. 74–102.

[16] Chapman, T. 2003. Lab automation and robotics: Automation on the move. *Nature*. 421, 6923 (2003), 661–666. DOI:https://doi.org/10.1038/421661a.

[17] Collins, S., Strasbourg, O.A. De, Harrower, N., Hodson, S., Jones, S., Centre, D.C., Laaksonen, L. and Mietchen, D. 2018. *FAIR Data Action Plan*.

[18] Crockford, D. 2017. *The JavaScript Object Notation (JSON) Data Interchange Format*. RFC-Document 8259 (https://tools.ietf.org/pdf/rfc8259.pdf).

[19] Daniels, J. 2009. Server virtualization architecture and implementation. *XRDS: Crossroads, The ACM Magazine for Students*. 16, 1 (2009), 8–12. DOI:https://doi.org/10.1145/1618588.1618592.

[20] Delaney, N.F., Echenique, J.I.R. and Marx, C.J. 2013. Clarity: An open-source manager for laboratory automation. *Journal of Laboratory Automation*. 18, 2 (2013), 171–177. DOI:https://doi.org/10.1177/2211068212460237.

[21] Dolgin, E. 2018. Labs on the cheap. *Nature*. 559, 7713 (2018), 291–293. DOI:https://doi.org/http://dx.doi.org/10.1038/d41586-018-05655-3.

[22] Eason, K.D. 1991. Ergonomic perspectives on advances in human-computer interaction. *Ergonomics*. 34, 6 (1991), 721–741. DOI:https://doi.org/10.1080/00140139108967347.

[23] Enste, U. and Mahnke, W. 2011. OPC Unified Architecture: Die nächste Stufe der Interoperabilität. *At-Automatisierungstechnik*. 59, 7 (2011), 397–405. DOI:https://doi.org/10.1524/auto.2011.0934.

[24] Extensible Markup Language (XML) 1.0, W3C Recommendation 10-February-1998: 1998. *https://www.w3.org/TR/1998/REC-xml-19980210*. Accessed: 2020-10-03.

[25] Feng, S., Caire, R., Cortazar, B., Turan, M., Wong, A. and Ozcan, A. 2014. Immunochromatographic diagnostic test analysis using google glass. *ACS Nano*. 8, 3 (2014), 3069–3079. DOI:https://doi.org/10.1021/nn500614k.

[26] Fielding, R.T. 2000. *Architectural Styles and the Design of Network-based Software Architectures*. Doctoral dissertation, University of California, Irvine (https://www.ics.uci.edu/~fielding/pubs/dissertation/rest_arch_style.htm).

[27] Fifty Years of RFCs: *https://tools.ietf.org/html/rfc8700*. Accessed: 2020-10-03.

[28] FreeOpcUa: Open Source C++ and Python OPC-UA Server and Client Libraries and Tools: *http://freeopcua.github.io/*. Accessed: 2020-10-01.

[29] Frey, J.G. 2004. Dark lab or smart lab: The challenges for 21st century laboratory software. *Organic Process Research and Development*. 8, 6 (2004), 1024–1035. DOI:https://doi.org/10.1021/op049895g.

[30] Gauglitz, G. 2018. Lab 4.0: SiLA or OPC UA. *Analytical and Bioanalytical Chemistry*. 410, 21 (2018), 5093–5094. DOI:https://doi.org/10.1007/s00216-018-1192-6.

[31] Grinias, J.P., Whitfield, J.T., Guetschow, E.D. and Kennedy, R.T. 2016. An inexpensive, open-source USB Arduino data acquisition device for chemical instrumentation. *Journal of Chemical Education*. 93, 7 (2016), 1316–1319. DOI:https://doi.org/10.1021/acs.jchemed.6b00262.

[32] gRPC, A high-performance, open source universal RPC framework: *https://grpc.io/*. Accessed: 2020-10-01.

[33] How JavaScript Works by Douglas Crockford: *https://www.json.org/json-en.html*. Accessed: 2020-10-03.

[34] Hu, G., Chen, L., Okerlund, J. and Shaer, O. 2015. Exploring the use of google glass in wet laboratories. *Conference on Human Factors in Computing Systems - Proceedings*. 18, (2015), 2103–2108. DOI:https://doi.org/10.1145/2702613.2732794.

[35] Huck-Fries, V., Wiegand, F., Klinker, K., Wiesche, M. and Krcmar, H. 2017. Datenbrillen in der Wartung: Evaluation verschiedener Eingabemodalitäten bei Servicetechnikern. *Informatik 2017*. (2017), 307–318. DOI:https://doi.org/10.18420/in2017.

[36] Hunt, C. 2002. *TCP/IP Network Administration: Help for Unix System Administrators*. O'Reilly Media, Inc.

[37] Hypertext Transfer Protocol version 2: *https://tools.ietf.org/html/draft-ietf-httpbis-http2-16*. Accessed: 2020-10-03.

[38] JSON-RPC: *https://www.jsonrpc.org/*. Accessed: 2020-10-02.

[39] Karnouskos, S., Bangemann, T. and Diedrich, C. 2009. *Integration of legacy devices in the future SOA-based factory*. IFAC.

[40] Kersken, S. 2013. *IT-Handbuch für Fachinformatiker, 6. Auflage*. Rheinwerk Verlag GmbH.

[41] Kesselmeier, H., Tschiersch, I. and Kutscha, S. 1997. The Re-Engineering of Complex Software Systems. *IFAC Proceedings Volumes*. 30, 24 (1997), 133–136. DOI:https://doi.org/10.1016/s1474-6670(17)42241-5.

[42] Kobayashi, M. 2016. More than moore. *Kyokai Joho Imeji Zasshi/Journal of the Institute of Image Information and Television Engineers*. 70, 3 (2016), 324–327. DOI:https://doi.org/10.3169/itej.70.324.

[43] Küchlin, W. and Weber, A. 2005. *Einführung in die Informatik*. Springer Berlin Heidelberg New York.

[44] Kutscha, S., Henning, K. and Kesselmeier, H. 1996. Der Task-Artifact-Cycle --- oder: Warum man Altsysteme nicht einfach wegwerfen sollte. *Softwarewartung und Reengineering: Erfahrungen und Entwicklungen*. F. Lehner, ed. Deutscher Universitätsverlag. 327–334.

[45] LADS – Laboratory Agnostic Device Standard, Ein neuer Standard für das smarte Labor: *https://www.spectaris.de/analysen-bio-und-labortechnik/vernetzte-laborgeraete/*. Accessed: 2020-10-01.

[46] LADS – Laboratory Agnostic Device Standard: *https://opcfoundation.org/markets-collaboration/lads/*. Accessed: 2020-10-01.

[47] Lütjohann, D.S., Jung, N. and Bräse, S. 2015. Open source life science automation: Design of experiments and data acquisition via "dial-a-device." *Chemometrics and Intelligent Laboratory Systems*. 144, (2015), 100–107. DOI:https://doi.org/10.1016/j.chemolab.2015.04.002.

[48] Merrick, P., Allen, S. and Lapp, J. 1999. XML remote procedure call (XML-RPC). US7028312B1. 1999.

[49] Miller, B.A., Nixon, T., Tai, C. and Wood, M.D. 2001. Home networking with Universal Plug and Play. *IEEE Communications Magazine*. 39, 12 (2001), 104–109. DOI:https://doi.org/10.1109/35.968819.

[50] Mons, B., Neylon, C., Velterop, J., Dumontier, M., da Silva Santos, L.O.B. and Wilkinson, M.D. 2017. Cloudy, increasingly FAIR; revisiting the FAIR Data guiding principles for the European Open Science Cloud. *Information Services & Use*. 37, 1 (Mar. 2017), 49–56. DOI:https://doi.org/10.3233/ISU-170824.

[51] Nationale Forschungsdateninfrastruktur – Ausschreibung 2019 für die Förderung von Konsortien: *https://www.dfg.de/foerderung/info_wissenschaft/2019/info_wissenschaft_19_37/index.html*. Accessed: 2020-10-01.

[52] Nelson, B.J. 1981. *Remote Procedure Call*. Palo Alto Research Center (http://www.bitsavers.org/pdf/xerox/parc/techReports/CSL-81-9_Remote_Procedure_Call.pdf).

[53] Not All RFCs are Standards: *https://tools.ietf.org/html/rfc1796*. Accessed: 2020-10-03.

[54] OPC Foundation Website: *https://opcfoundation.org*. Accessed: 2020-10-01.

[55] OPC UA ANSI-C Implementation: *https://open62541.org/*. Accessed: 2020-10-01.

[56] opcua4j, open source implementation of an opc ua server in java: *https://code.google.com/archive/p/opcua4j/*. Accessed: 2020-10-01.

[57] Porr, M., Lange, F., Marquard, D., Niemeyer, L., Lindner, P., Scheper, T. and Beutel, S. Implementing a Digital Infrastructure for the Lab Using a Central Laboratory Server and the SiLA2 Communication Standard. *Engineering in Life Sciences*.

[58] Porr, M., Marquard, D., Stanislawski, N., Austerjost, J., Russo, M., Bungers, S., Klimmt, C., Scheper, T., Beutel, S. and Lindner, P. 2019. smartLAB – Working Interactively in a Digitalized Laboratory Environment. *Chemie-Ingenieur-Technik*. 91, 3 (2019). DOI:https://doi.org/10.1002/cite.201800090.

[59] Porr, M., Schwarz, S., Lange, F., Niemeyer, L., Hentrop, T., Marquard, D., Lindner, P., Scheper, T. and Beutel, S. 2020. Source Files for tci-gatewaymodule. *Mendeley Data Repository*. (2020). DOI:https://doi.org/10.17632/fvccdd6r4f.1.

[60] Porr, M., Schwarz, S., Lange, F., Niemeyer, L., Marquard, D., Lindner, P., Scheper, T. and Beutel, S. 2020. Bringing IoT to the lab: sila2 and open-source-powered gateway module for integrating legacy devices into the digital laboratory. *HardwareX*. (2020), e00118. DOI:https://doi.org/10.1016/j.ohx.2020.e00118.

[61] Portnoy, M. 2016. *Virtualization Essentials, 2nd Edition*.

[62] Preece, J.J., Rogers, Y., Sharp, H.C., Benyon, D., Holland, S. and Carey, T.T. 1994. *Human-Computer Interaction*.

[63] Protocol Buffers: *https://developers.google.com/protocol-buffers*. Accessed: 2020-10-03.

[64] Reif, R. and Günthner, W.A. 2009. Pick-by-vision: augmented reality supported order picking. *Visual Computer*. 25, 5–7 (2009), 461–467. DOI:https://doi.org/10.1007/s00371-009-0348-y.

[65] Roth, A., Jopp, R., Schäfer, R. and Kramer, G.W. 2006. Automated Generation of AnIML Documents by Analytical Instruments. *JALA - Journal of the Association for Laboratory Automation*. 11, 4 (2006), 247–253. DOI:https://doi.org/10.1016/j.jala.2006.05.013.

[66] Schäfer, B.A., Poetz, D. and Kramer, G.W. 2004. Documenting Laboratory Workflows Using the Analytical Information Markup Language. *Journal of Laboratory Automation*. 9, 6 (2004), 375–381. DOI:https://doi.org/10.1016/j.jala.2004.10.003.

[67] Schmid, I. and Aschoff, J. 2017. A scalable software framework for data integration in bioprocess development. *Engineering in Life Sciences*. 17, 11 (2017), 1159–1165. DOI:https://doi.org/10.1002/elsc.201600008.

[68] selfHtml Wiki: *https://wiki.selfhtml.org*. Accessed: 2020-10-03.

[69] sila_tecan: 2020. *https://gitlab.com/SiLA2/vendors/sila_tecan*. Accessed: 2020-07-07.

[70] SiLA 2 Part (C) - Standard Features Index: 2019. *https://drive.google.com/file/d/1dqQTqRN6vyJy6KBISCsnV_qte_kNh62l/view*. Accessed: 2020-07-07.

[71] SiLA 2 Part (A) - Overview, Concepts and Core Specification: 2019. *https://drive.google.com/file/d/1QWrSD4-YBMwT9HTBzJe3TlBbJSGqq_LC/view*. Accessed: 2020-07-07.

[72] SiLA 2 Part (B) - Mapping Specification: 2019. *https://drive.google.com/file/d/1a9XJnQUHysW6DQGz4j2m1zngLX-pakxo/view*. Accessed: 2020-07-07.

[73] SiLA Rapid Integration: *https://sila-standard.com/*. Accessed: 2020-10-01.

[74] SiLA2 - official software repository: 2019. *https://gitlab.com/SiLA2*. Accessed: 2020-07-07.

[75] Solle, D. 2020. Be FAIR to your data. *Analytical and Bioanalytical Chemistry*. (2020). DOI:https://doi.org/10.1007/s00216-020-02526-7.

[76] Stephan, C., Kohl, M., Turewicz, M., Podwojski, K., Meyer, H.E. and Eisenacher, M. 2010. Using laboratory information management systems as central part of a proteomics data workflow. *Proteomics*. 10, 6 (2010), 1230–1249. DOI:https://doi.org/10.1002/pmic.200900420.

[77] Wegner, P. 1996. Interoperability. *ACM Computing Surveys*. 28, 1 (1996), 285–287. DOI:https://doi.org/10.1145/234313.234424.

[78] Weiser, M. 1991. The Computer for the 21st Century. *Osaka Journal of Mathematics*. 11, 3 (1991), 577–586.

[79] Welcome to AnIML: *https://animl.org/*. Accessed: 2020-10-02.

[80] What is HDF5: *https://portal.hdfgroup.org/display/knowledge/What+is+HDF5*. Accessed: 2020-10-02.

[81] Wilkinson, M.D. et al. 2019. Addendum: The FAIR Guiding Principles for scientific data management and stewardship. *Scientific Data*. 6, 1 (Dec. 2019), 6. DOI:https://doi.org/10.1038/s41597-019-0009-6.

[82] Wilkinson, M.D. et al. 2016. Comment: The FAIR Guiding Principles for scientific data management and stewardship. *Scientific Data*. 3, (2016), 1–9. DOI:https://doi.org/10.1038/sdata.2016.18.

[83] Wilkinson, M.D. et al. 2016. The FAIR Guiding Principles for scientific data management and stewardship. *Scientific Data*. 3, 1 (Dec. 2016), 160018. DOI:https://doi.org/10.1038/sdata.2016.18.

[84] Zhang, P. and Li, N. 2004. An assessment of human–computer interaction research in management information systems: topics and methods. *Computers in Human Behavior*. 20, 2 (2004), 125–147. DOI:https://doi.org/https://doi.org/10.1016/j.chb.2003.10.011.

ANHÄNGE

A ABBILDUNGSVERZEICHNISS

B UNTERSTÜTZENDE ARBEITEN

chronologisch

- *Bachelorarbeit*
  Laura Niemeyer: „Geräteintegration für die Labordigitalisierung“, 2018
- *Bachelorarbeit*
  Michelle Angelica Djuari: „Digitale Integration mittels SiLA2 und FAIR-Data anhand einer kontinuierlichen Chromatographieanlage“, 2020
- *Bachelorarbeit*
  Karolina Sikora: „Ausarbeitung, Umsetzung und Evaluation von digitalen Laborabläufen“, 2020
- *Masterarbeit*
  Tessa Habich: „Entwicklung von Konzepten und Methoden zur Integration von Downstream-Prozesseinheiten im digitalisierten Labor“, 2020

C VERÖFFENTLICHUNGEN

Umgekehrt chronologisch, peer-reviewed

- **Marc Porr (80 %)**, Ferdinand Lange, Daniel Marquard, Laura Niemeyer, Patrick Lindner, Thomas Scheper, Sascha Beutel: *„Implementing a Digital Infrastructure for the Lab Using a Central Laboratory Server and the SiLA2 Communication Standard"* Engineering in Life Sciences (2020), http://doi.org/10.1002/elsc.202000053
- Daniel Marquard, **Marc Porr (20 %)**, Ferdinand Lange, Jonas Austerjost, Sascha Beutel: „Smartglasses im Labor – Werkzeug oder Spielzeug?" Chemie in unserer Zeit (2020), https://doi.org/10.1002/ciuz.202000022
- **Marc Porr (75 %)**, Sebastian Schwarz, Ferdinand Lange, Laura Niemeyer, Thorleif Hentrop, Daniel Marquard, Patrick Lindner, Thomas Scheper, Sascha Beutel: *„Bringing IoT to the Lab: SiLA2 and Open-Source-Powered Gateway Module for Integrating Legacy Devices into the Digital Laboratory"* HardwareX (2020), https://doi.org/10.1016/j.ohx.2020.e00118
- **Marc Porr (80 %)**, Daniel Marquard, Nils Stanislawski, Jonas Austerjost, Mario Russo, Simon Bungers, Christoph Klimmt, Thomas Scheper, Sascha Beutel, Patrick Lindner: „smartLAB – Interaktives Arbeiten in digitalisierter Laborumgebung" Chemie Ingenieur Technik (2019), https://doi.org/10.1002/cite.201800090
- Jonas Austerjost, Malte Bargholz, **Marc Porr (10 %)**, Dominik Geier, Thomas Becker, Daniel Marquard, Patrick Lindner, Thomas Scheper, Sascha Beutel: *„A flexible IT infrastructure for the integration of smartglasses into the brewing laboratory as a digital support for standard analysis workflows"* BrewingScience (2019), https://doi.org/10.23763/BrSc18-20austerjost
- Jonas Austerjost, **Marc Porr (10 %)**, Noah Riedel, Dominik Geier, Thomas Becker, Thomas Scheper, Daniel Marquard, Patrick Lindner, Sascha Beutel: *„Introducing a Virtual Assistant to the Lab: A Voice User Interface for the Intuitive Control of Laboratory Instruments"* SLAS Technology (2018), https://doi.org/10.1177/2472630318788040

D KONFERENZBEITRÄGE UND WORKSHOPS

Umgekehrt chronologisch

- *Vortrag*
  Marc Porr: „smartLAB: Digital, integrated and automated; the future of the LIMS and how to work with it", Future LABS Live, Basel (Schweiz), 2020
- *Vortrag*
  Johannes Hemmerich, **Marc Porr**, Nikolas von den Eichen: „Digitalization in Industrial Biotechnology - Challenges of miniaturization, automation and interfaces", 10. ProcessNet-Jahrestagung und 34. DECHEMA-Jahrestagung der Biotechnologen, Aachen (Deutschland), 2020
- *Poster*
  Marc Porr, Daniel Marquard, Ferdinand Lange, Jonas Austerjost, Kirsten McVean, Andrea Herrmann, Chrisitan Maeß, Patrick Lindner, Thomas Scheper, Sascha Beutel: „Digital image analysis in the lab of the future", 10. ProcessNet-Jahrestagung und 34. DECHEMA-Jahrestagung der Biotechnologen, Aachen (Deutschland), 2020
- *Poster*
  Laura Niemeyer, **Marc Porr**, Chantal Brämer, Jonas Austerjost, Ferdinand Lange, Daniel Marquard, Thomas Scheper, Sascha Beutel,Patrick Lindner: „Digital integration of a periodic counter-current chromatography system", DECHEMA Himmelfahrtstagung, Hamburg (Deutschland), 2019
- *Workshop*
  SiLA 2 Hackathon #17, SiLA Consortium, Wiesbaden (Deutschland), 2019
- *Workshop*
  7. Expertenworkshop: „Datenbrillen – Aktueller Stand von Forschung und Umsetzung sowie zukünftiger Entwicklungsrichtungen", Bundesanstalt für Arbeitsschutz und Arbeitsmedizin, Dortmund (Deutschland), 2018

E LEBENSLAUF

Umgekehrt chronologisch

Persönliche Angaben

Name	Marc Julian Porr
Geburtsdaten	25.03.1992 in Remscheid
Staatsangehörigkeit	deutsch

Beruflicher Werdegang

Seit November 2020	**Anstellung als Software Architekt** *KISTERS AG, Oldenburg*
Dez. 2017 – Okt. 2020	**Anstellung als wissenschaftlicher Mitarbeiter** *Institut für Technische Chemie, Leibniz Universität Hannover*
Sept. 2014 – März 2015	**Anstellung als Process Worker** *Zip Industries (Sydney, Australien)*
Feb. 2013 – Okt. 2017	**Anstellung als studentische Hilfskraft** *Institut für Technische Chemie, Leibniz Universität Hannover*

Ausbildung

2017 – heute	**Promotionsstudium, Angestrebter Abschluss: Dr. rer. nat.** *Institut für Technische Chemie, Leibniz Universität Hannover*
2015 – 2017	**Master-Studium Life Science** (Abschlussnote: 1,2) *Institut für Technische Chemie, Leibniz Universität Hannover*
2011 – 2015	**Bachelor-Studium Life Science** (Abschlussnote: 1,2) *Institut für Technische Chemie, Leibniz Universität Hannover*
2010 – 2011	**Zivildienst** *Diakoniewerk Jerusalem, Bad Bevensen*
1997 – 2010	**Schulausbildung** (Abiturnote: 1,4) *Lessing Gymnasium Uelzen,* *Orientierungsstufe Ebstorf,* *Mauritius Grundschule Ebstorf,* *Grundschule Dünn*