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

A R T I C L E I N F O

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

A B S T R A C T

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/© 2020 The Author(s). Published by Elsevier Ltd.

* (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.

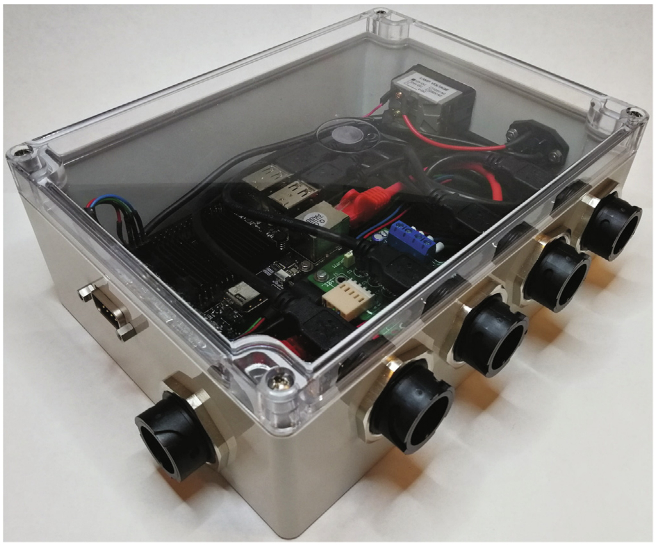


Fig. 1. The gateway module fully assembled in an IP65-certified splash-water proof casing.

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

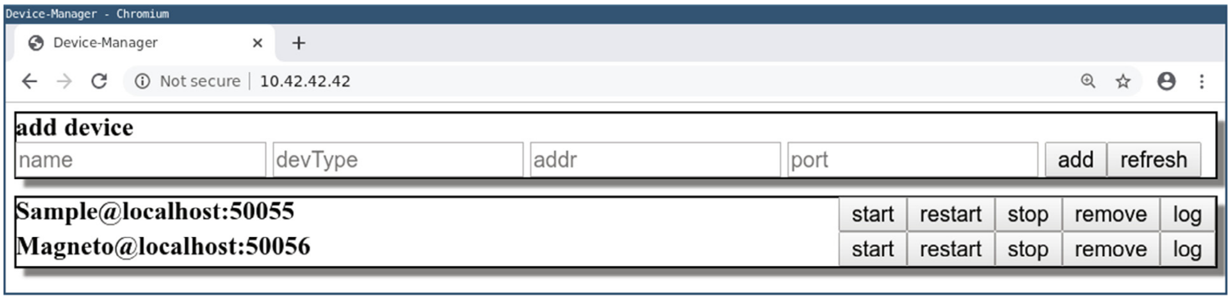
For applications where spillage of liquids may occur (as usually in chemical/biotechnological labs) or dust exposure is a problem, a version with a completely enclosed casing is presented. All components that are in contact with the box's exterior are certified as IP65 or higher according to IEC standard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



add device

name	devType	addr	port	add	refresh

Sample@localhost:50055	start	restart	stop	remove	log
Magneto@localhost:50056	start	restart	stop	remove	log

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

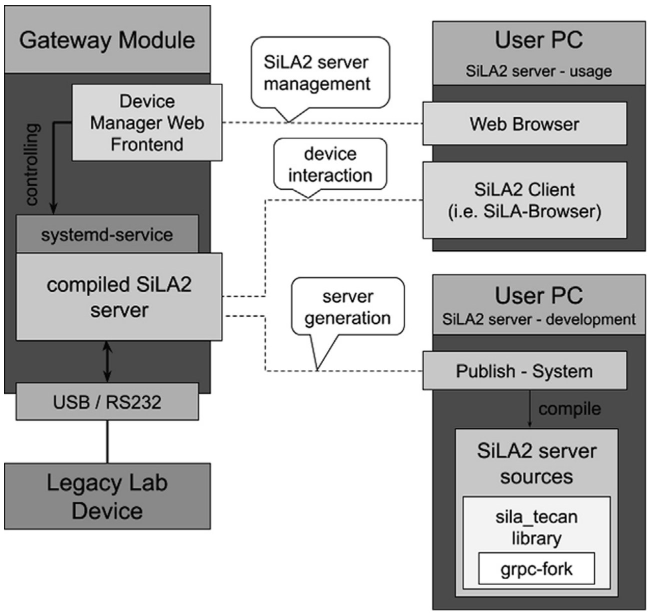


Fig. 3. Overview of the software architecture in the configuration presented. The physical location of all software components is highlighted. The Device-Manager web frontend is running directly on the gateway module in this example. During development the SiLA2 server gets published to the gateway module 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_M223Gcover.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-18.04.3-3.16odroid_systemimage20200311.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/silatecan.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].

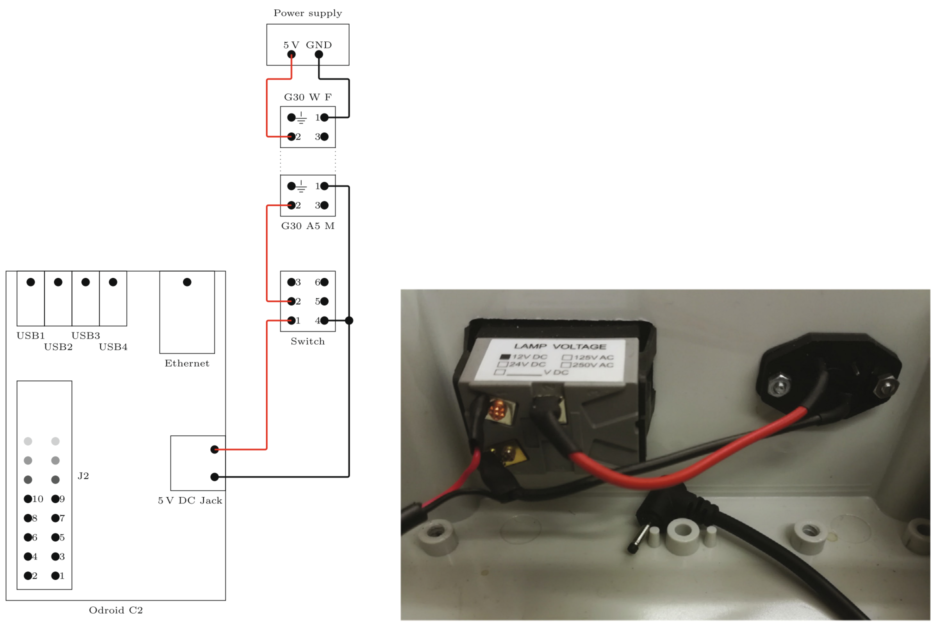
5.3.1. Minimal and RS232 version power supply assembly

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

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

```
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"
```

Where:

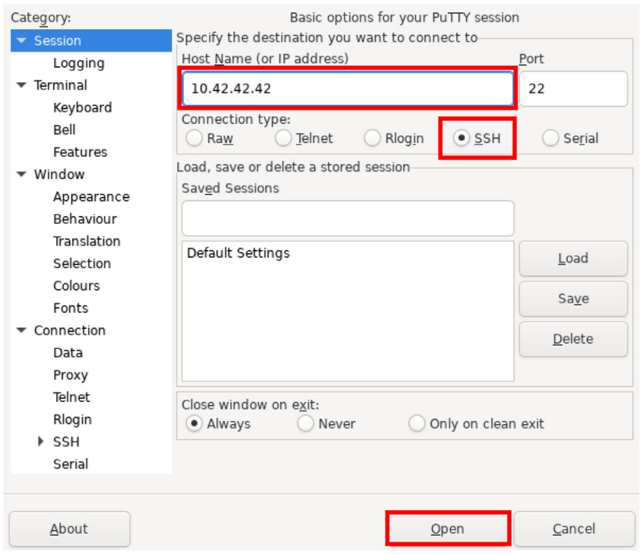
name would be the name of your new connection (just choose any you like)
x.x.x.x/yy would be the desired IP address and subnet mask in CIDR notation
z.z.z.z would be the IP address of your gateway (omit the gw4 parameter if you do not have one)
a.a.a.a,b.b.b.b would be the addresses of the DNS servers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


Category:
Basic options for your PuTTY session
Specify the destination you want to connect to
Host Name (or IP address): 10.42.42.42
Port: 22
Connection type: Raw Telnet Rlogin [✓ SSH] Serial
Load, save or delete a stored session
Saved Sessions
Default Settings
Close window on exit: [✓ Always] Never Only on clean exit

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!

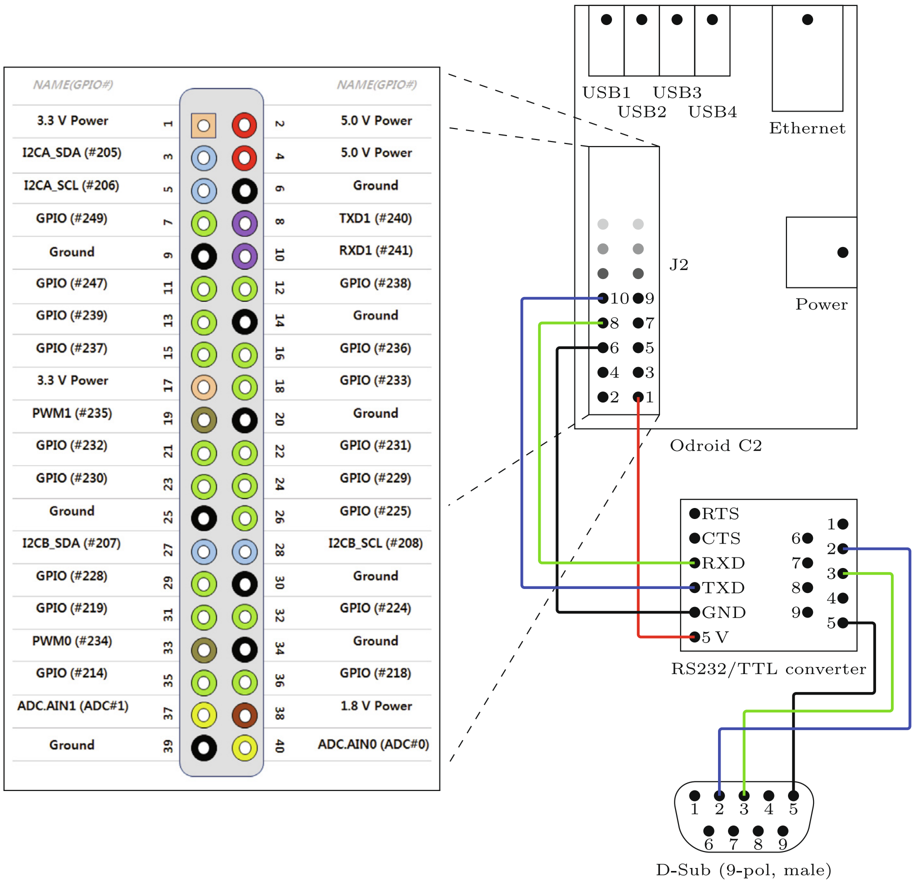
Caution: When soldering the converter chip please limit its heat uptake to a minimum to protect the circuits inside.

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.

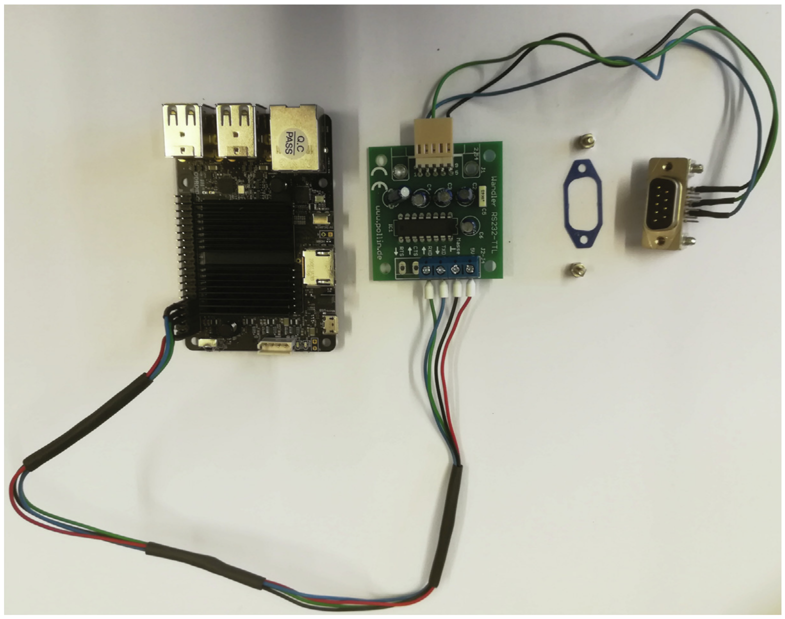
Caution: Do not solder a male D-SUB connector directly to the RS232 converter board!

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.

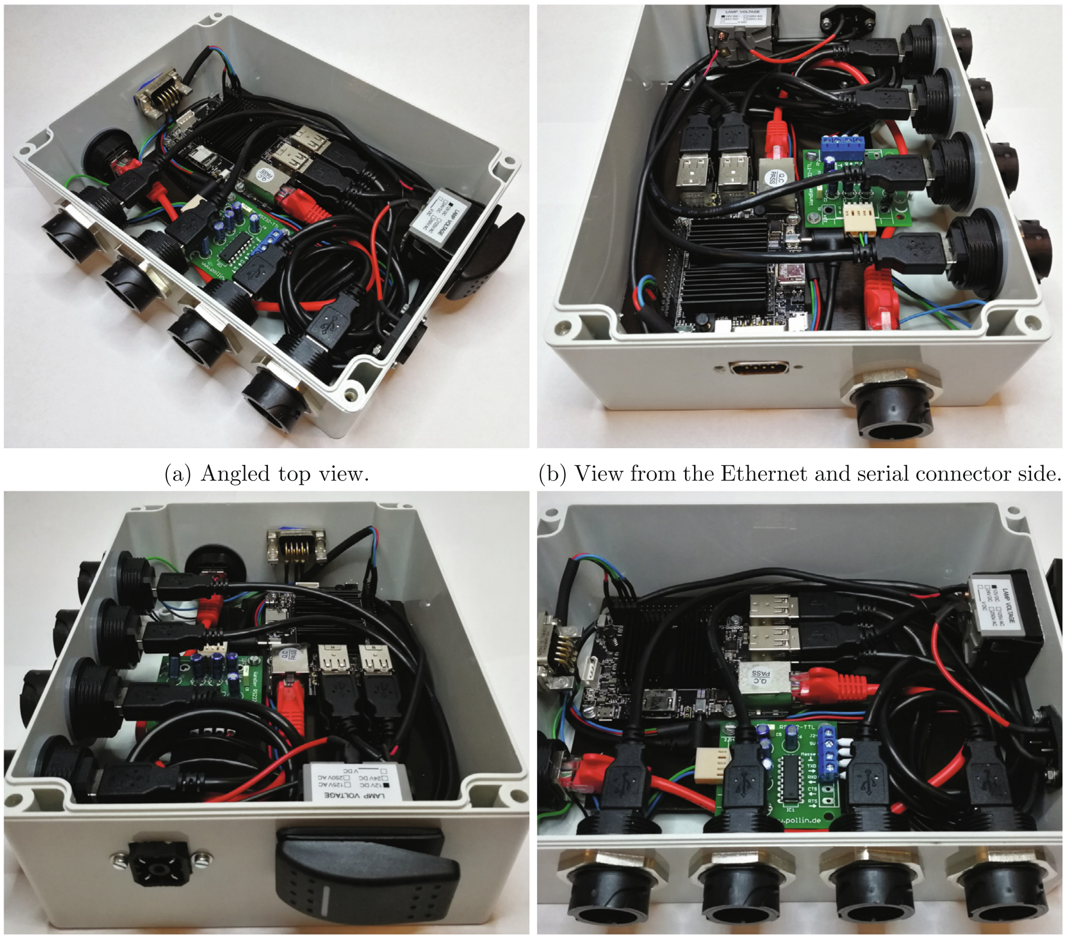


(a) Wiring diagram for connecting the RS232/TTL converter setup.



(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.

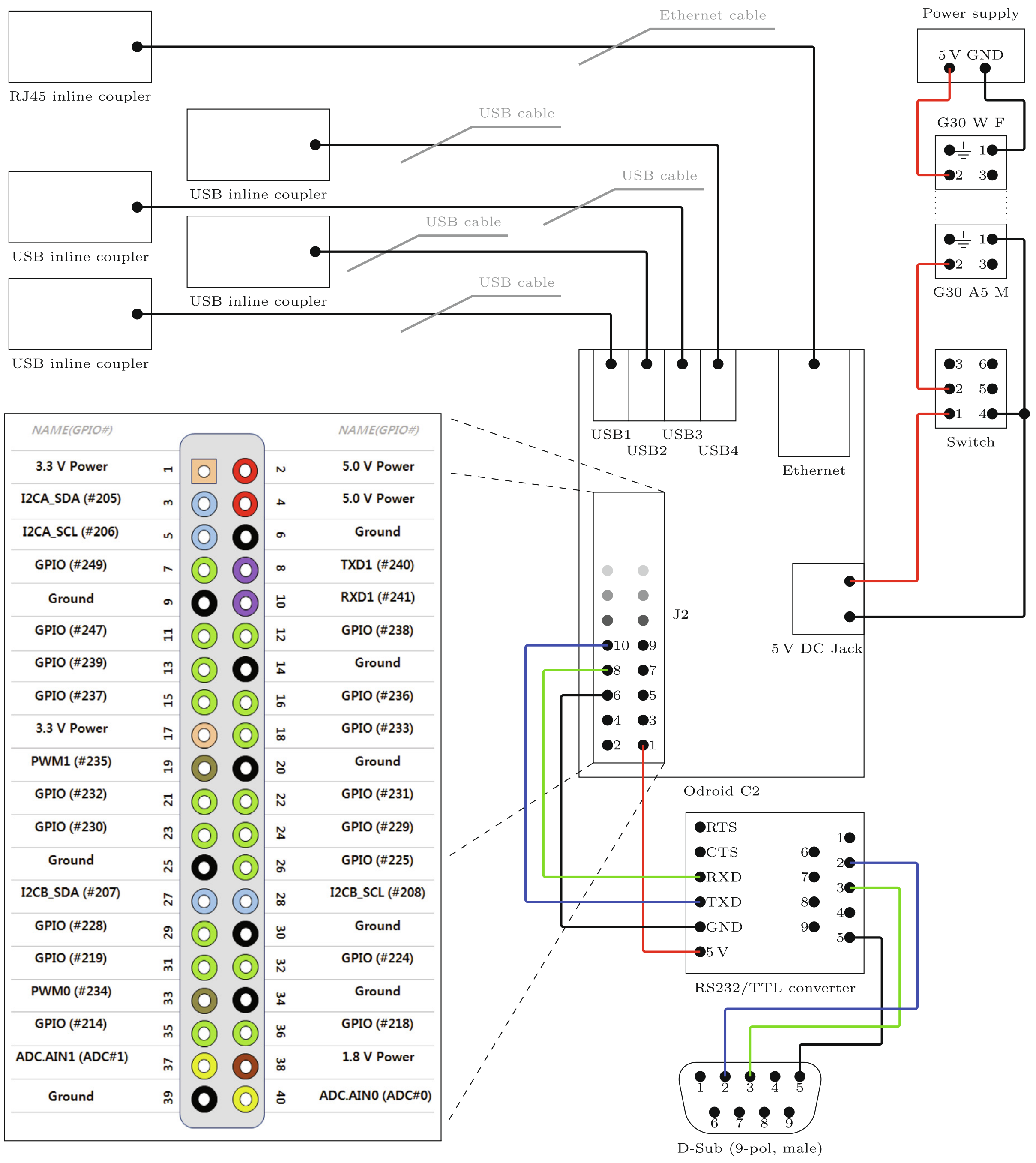


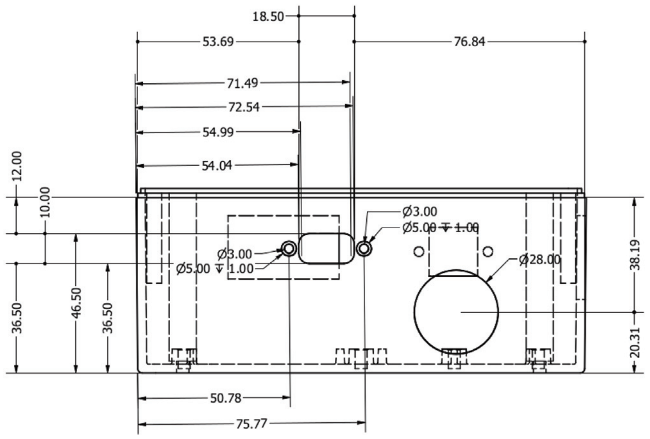
Fig. 7. Complete wiring diagram for the IP65 version (Source of J2-header pinout drawing: [18]).

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).

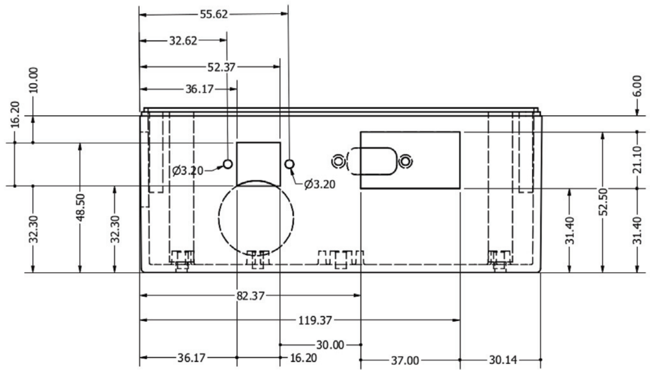
Connect the USB and RJ45 Couplers as can be seen in Fig. 13. Use the 0.5 m USB- or 0.25 m Ethernet-cables.

6. Operation instructions

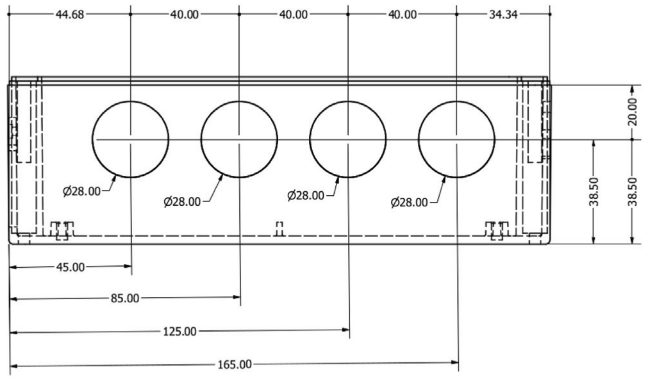
In this section the setup and usage of the gateway module is described. If you want to start off with the pre-prepared system image to have a ready-to-operate gateway module with some example SiLA2 servers already installed, please follow the instructions in Section 6.2.



(a) Ethernet and serial connector side.



(b) Power connector side.



(c) USB connector side.

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.

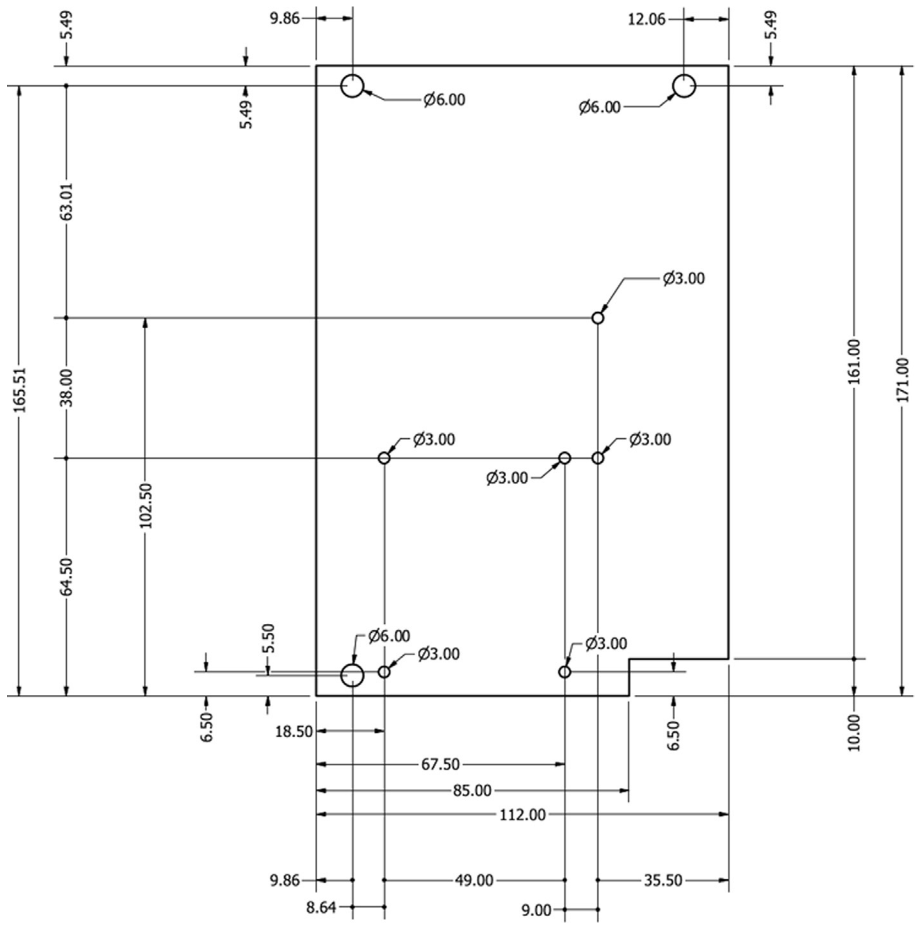


Fig. 9. Template for cutting and drilling the baseplate out of Pertinax Hard Paper Plate.

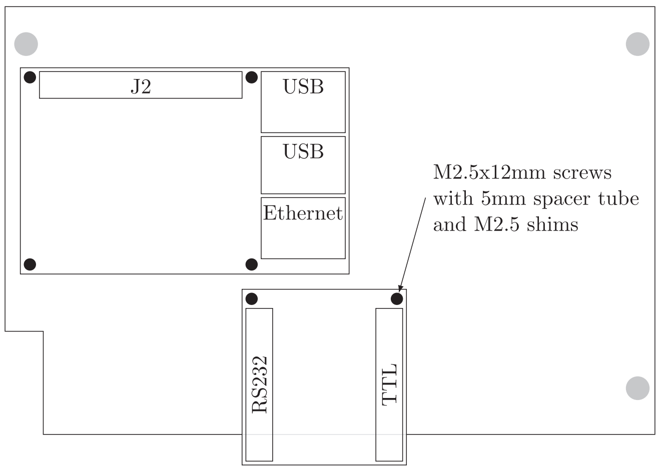


Fig. 10. Sketch of ODroid board and serial converter board positions on the basepl.

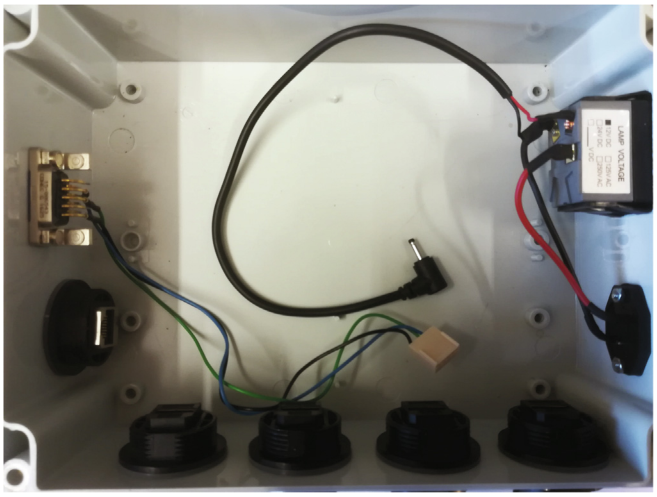


Fig. 11. RJ45 Coupler, four USB Couplers, Belden Receptacle for power connection, Rocker Switch Assembly and the D-SUB Connector mounted in the Bopla M 223 G enclosure.

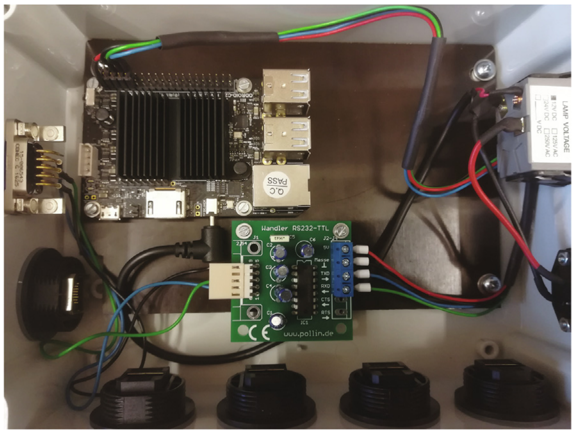


Fig. 12. The assembled baseplate screwed inside the box and connected to the serial converter and power supply.

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.

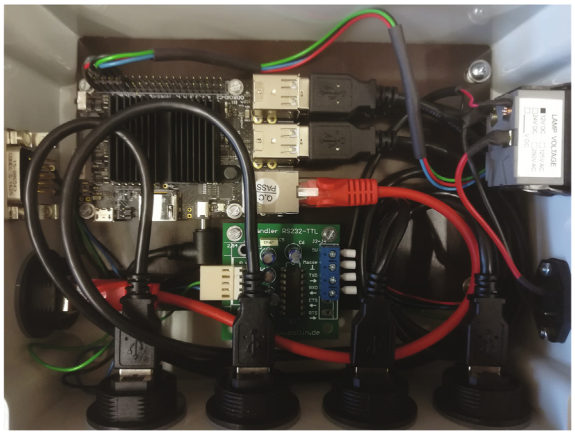


Fig. 13. The USB and Ethernet cables connected to the couplers.

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

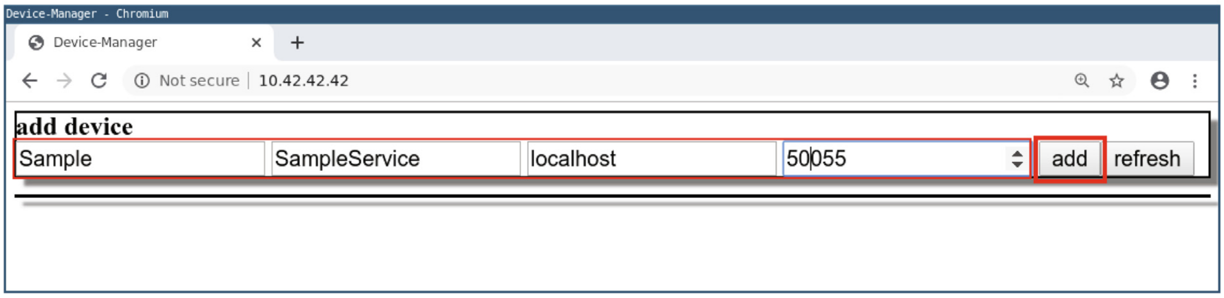
Now take a close look at sila_tecan/Examples/SampleServer and its readme [34] as it explains the steps to create a new server in detail. Follow these steps to create your new server. Make sure to create a dotnet core application as it will not run on the Linux system of the gateway module when you develop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

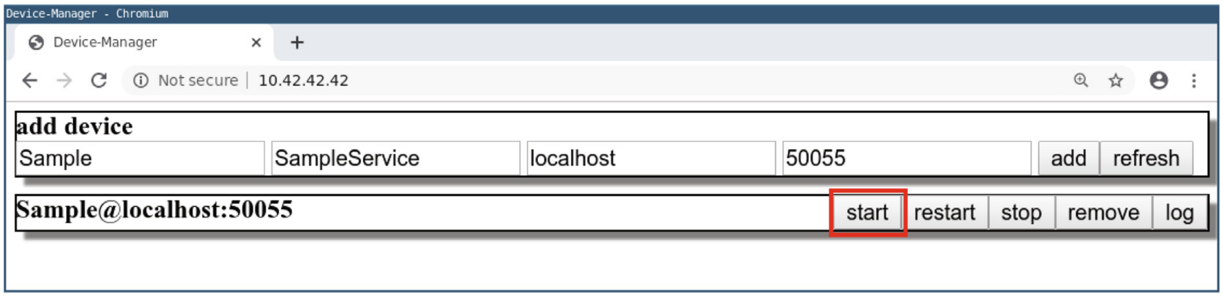
6.3.5. Publish the server to the gateway module

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*.

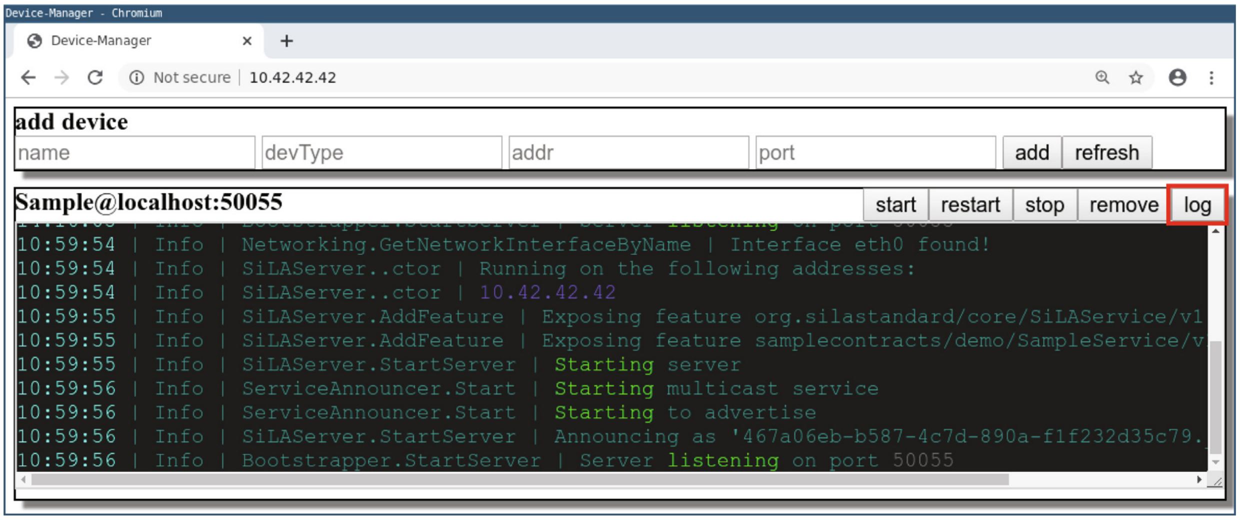
```
me@desktop:~$ git clone https://gitlab.uni-hannover.de/tci-gateway-module/gateway-publish
me@desktop:~$ cd gateway-publish
me@desktop:~/gateway-publish$ ./publish.sh 10.42.42.42 ../sila_tecan/Examples/SampleServer/SampleService
    ↪ /SampleService.csproj
```



(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.



```
10:59:54 | Info | Networking.GetNetworkInterfaceByName | Interface eth0 found!
10:59:54 | Info | SiLAServer..ctor | Running on the following addresses:
10:59:54 | Info | SiLAServer..ctor | 10.42.42.42
10:59:55 | Info | SiLAServer.AddFeature | Exposing feature org.silastandard/core/SiLAService/v1
10:59:55 | Info | SiLAServer.AddFeature | Exposing feature samplecontracts/demo/SampleService/v
10:59:55 | Info | SiLAServer.StartServer | Starting server
10:59:56 | Info | ServiceAnnouncer.Start | Starting multicast service
10:59:56 | Info | ServiceAnnouncer.Start | Starting to advertise
10:59:56 | Info | SiLAServer.StartServer | Announcing as '467a06eb-b587-4c7d-890a-f1f232d35c79.
10:59:56 | Info | Bootstrapper.StartServer | Server listening on port 50055
```

(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
```

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].

[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).

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 preconfigured 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 dotnet 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.

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.

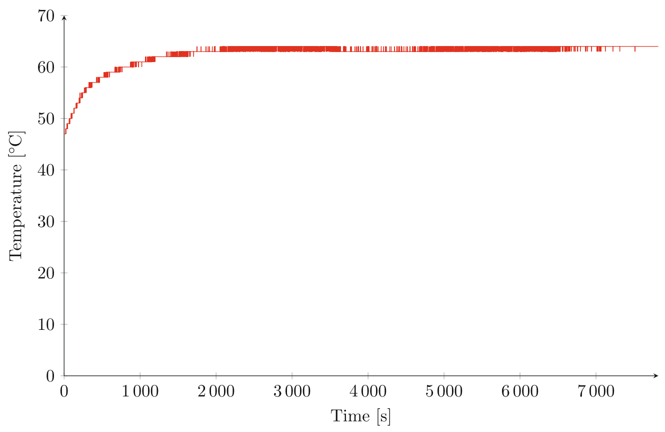


Fig. 15. Temperature readings from the ODroids internal thermal sensor during stress test.

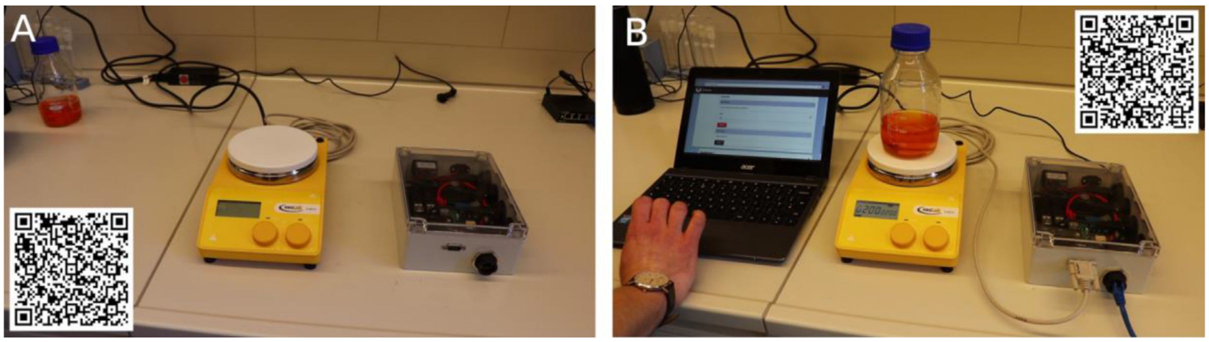


Fig. 16. Usecase examples of the gateway module used to act as a SiLA2 server for a *neoLab D-6010* magnetic stirrer. A: Preparation of the module (Link to video: https://data.mendeley.com/datasets/fvccdd6r4f/1/files/6eb9763f-b6e0-4a8d-a01b-7c916958a7f4/magnetosetup.mp4?dl=1), B: Presentation of the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

7.3. 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)..