TAMPEREEN TEKNILLINEN YLIOPISTO
TAMPERE UNIVERSITY OF TECHNOLOGY

ARI STJERNA

DEPLOYMENT OF CLOUD BASED PLATFORMS FOR PROCESS DATA GATHERING AND VISUALIZATION IN PRODUCTION AUTOMATION

Master of Science thesis

Examiners: prof. José L. Martinez Lastra, Dr. Jani Jokinen
Examiner and topic approved by the Faculty Council of the Faculty of Engineering Sciences
on 1th March 2017

ABSTRACT

ARI STJERNA: Deployment of Cloud Based Platforms for Process Data Gathering and Visualization in Production Automation
Tampere University of technology
Master of Science Thesis, 111 pages, 17 Appendix pages
March 2017
Master's Degree Programme in Automation Technology
Major: Factory Automation
Examiners: Professor José L. Martinez Lastra, Senior Research Fellow Jani Jokinen

Keywords: cloud computing, REST interface, Internet-of-Things, future production automation monitoring

New developments at the field of factory information systems and resource allocation solutions are constantly taken into practice within the field of manufacturing and production. Customers are turning their vision for more customized products and requesting further monitoring possibilities for the product itself, for its manufacturing and for its delivery. Similar paradigm change is taking place within the companies' departments and between the clusters of manufacturing stakeholders. Modern cloud based tools are providing the means for gaining these objectives.

Technology evolved from parallel, grid and distributed computing; at present cited as Cloud computing is one key future paradigm in factory and production automation. Regardless of the terminology still settling, in multiple occasions cloud computing is used term when referring to cloud services or cloud resources. Cloud technology is furthermore understood as resources located outside individual entities premises. These resources are pieces of functionalities for gaining overall performance of the designed system and so worth such an architectural style is referred as Resource-Oriented Architecture (ROA). Most prominent connection method for combining the resources is a communication via REST (Representational State Transfer) based interfaces. When comping cloud resources with internet connected devices technology, Internet-of-Things (IoT) and furthermore IoT Dashboards for creating user interfaces, substantial benefits can be gained. These benefits include shorter lead-time for user interface development, process data gathering and production monitoring at higher abstract level.

This Master's Thesis takes a study for modern cloud computing resources and IoT Dashboards technologies for gaining process monitoring capabilities able to be used in the field of university research. During the thesis work, an alternative user group is kept in mind. Deploying similar methods for private production companies manufacturing environments. Additionally, field of Additive Manufacturing (AM) and one of its subcategory Direct Energy Deposition Method (DED) is detailed for gaining comprehension over the process monitoring needs, laying in the questioned manufacturing method. Finally, an implementation is developed for monitoring Tampere University of Technology Direct Energy Deposition method manufacturing environment research cell process both in real-time and gathering the process data for later reviewing. These functionalities are gained by harnessing cloud based infrastructures and resources.

TIIVISTELMÄ

ARI STJERNA: Pilvipalvelupohjaisten alustojen hyödyntäminen tuotantoautomaation prosessidatan keräyksessä ja visualisoinnissa
Tampereen teknillinen yliopisto
Diplomityö, 111 sivua, 17 liitesivua
Maaliskuu 2017
Automaatiotekniikan diplomi-insinöörin tutkinto-ohjelma
Pääaine: Factory Automation
Tarkastajat: Professori José L. Martinez Lastra, Yliopistotutkija Jani Jokinen

Avainsanat: pilvipalvelut, REST rajapinnat, esineiden internet, tulevaisuuden tuotantoautomaation monitorointi

Sekä teollisuuden valmistuksessa että -tuotannossa otetaan jatkuvasti käyttöön teollisuusautomaation informaatio järjestelmissä sekä käytettävien resurssien allokoinneissa tapahtuvien kehitys askeleiden mukanaan tuomia uusia ominaisuuksia. Samalla kuluttajat ovat lisänneet kiinnostustaan laajempiin monitorointi ominaisuuksiin aina tuotteiden valmistuksesta, toimitukseen saakka. Vastaavanlainen muutos on tapahtumassa teollisuuden piirissä, sekä yritysten sisällä, että yritysten välillä. Valmistavan teollisuuden toimijat ovat entuudestaan muodostaneet ympärilleen alihankkijoiden yhteenliittymiä eli klustereita. Nyt näiden klustereiden sisällä ryhdytään vaatimaan parempia monitorointi kyvykkyyksiä. Modernit pilvipalveluihin perustuvat työkalut mahdollistavat näiden tavoitteiden saavuttamisen.

Verkko-, rinnakkaisuus- sekä hajautetun laskentatekniikan uusin kehitysmuoto nimeltä pilvilaskenta sekä laajemmin pilvipalvelut tulevat olemaan teollisuusautomaation tulevaisuuden avaintekijöitä. Näiden palveluiden yhteydessä käytettävä terminologia on vielä vakiintumassa. Monesti puhuttaessa pilvilaskennasta, tarkoitetaan palveluita ja resursseja jotka sijaitsevat toimijoiden omien toimipisteiden ulkopuolella, kolmannen osapuolen hallitsemina. Liitettäessä näitä resursseja yhteen resurssipohjaisten arkkitehtuurien menetelmien, saadaan aikaan uusia toiminnollisuuksia toteuttavia järjestelmiä. Eräs lupaavin kommunikointitekniikka eri palveluita yhdistettäessä on REST rajapintoihin perustuvat menetelmät. Esineiden internet sekä tarkemmin mainittuna esineiden internet alustojen käyttöliittymätyökalut luovat pilvipalvelu resursseille lisäominaisuuksia käyttöliittymien luonnissa. Näitä ominaisuuksia ovat lyhyemmät läpimenoajat suunnittelussa, sekä prosessien monitoroinnin helpottuminen.

Tämä diplomityö perehtyy moderneihin pilvipalvelu resursseihin sekä esineiden internet käyttöliittymä työkaluihin, tavoitteena hyödyntää näitä yliopistotason kokeellisten tutkimusjärjestelmien monitoroinnissa. Työn aikana pidettiin mielessä myös mahdollisuus hyödyntää vastaavia menetelmiä yksityisen sektorin valmistavan tuotannon järjestelmissä. Lisäksi työssä perehdytään ainetta lisäävän valmistusmenetelmän alalajiin, suorakerrostuksen vaativiin monitorointi ominaisuuksiin. Työn käytännön osuudessa toteutettiin Tampereen teknillisen yliopiston suorakerrostus valmistuksen tutkimusympäristöön monitorointi sovellus. Sovelluksen avulla monitoroidaan kyseisen tuotantojärjestelmän prosessia sekä reaaliajassa että keräten tarkempaa prosessidataa myöhemmin tarkasteltavaksi. Nämä ominaisuudet toteutettiin pilvipalveluihin perustuvilla infrastruktuureilla sekä resursseilla.

PREFACE

This thesis is the final goal on my personal endeavor for reaching my Master of Science degree. After I started the project as a Bachelor of Engineering for four years ago, it has been a long road of time-consuming assignments and patience-taking exams. Now, when finalizing my thesis I would like to express my gratitude to several persons.

First, I express my gratitude to my thesis mentors Professor José L. Martinez Lastra and Senior Research Fellow Jani Jokinen. Additionally, I like to recognize the input of Research Manager Jorma Vihinen and Project Manager Jyrki Latokartano for planting the seed and developing the idea and topic of my thesis.

My special thanks goes to Mr. Oskari Hakaluoto, founder of Roboco Co. He originally developed the History Data gathering text file parsing program for our disposal. Later used in robot controller in the implementation. I also thank him for his extensive consult and aid for working with the ABB robot in the application environment. His knowledge was and still is, indispensable.

Initial step for reaching the start point of my Master of Science studies, was taken almost a decade ago in 2008 when I graduated for Bachler of Engineering. In the preface of my Bachelors thesis, I sent my compliments to my parents for their never-ending support. That support still exists. However, now at the end of this study project it is time to reach out for other closest persons.

Nothing mentioned above can compete with the gratitude that I owe to my fiancée Jenna. She never lost fate in my perseverance and commitment for reaching the dream of Master of Science degree. I apologize for these four passed years. Years, which can never be reclaimed. I could not have done without her. One more person to be remembered. Our daughter who is now three months old. She will not remember the time when her dad was pushing the hours for finishing with his thesis. Nonetheless, last special greeting to you, Sandra.

Tampere, 14th March 2017

Ari Stjerna

CONTENTS

LIST OF FIGURES

LIST OF TABLES

LIST OF SYMBOLS AND ABBREVIATIONS

ACL	Access Control List
AIOTI	Alliance for IoT Innovation
AJAX	Asynchronous JavaScript And XML
AM	Additive Manufacturing
AMI	Amazon Machine Image
API	Application Protocol Interface
AWS	Amazon Web Services
BI	Business Intelligence
BPEL	Business Process Execution Language
CaaS	Communications as Service
CAD	Computer-Aided Design
CAM	Computer-Aided Manufacturing
CAM	Computer Aided Manufacturing
CAN	Control Area Network
CIA	Confidentially, Integrity and Availability
CIDR	Classless Inter-Domain Routing blocks
CIP	Common Industrial Protocol
CLI	Command Line Interface
CMT	Cold Metal Transfer
CNC	Computer Numerical Control
CPS	Cyber-Physical Systems
CSS	Cascading Style Sheets
CTO	Configure-To-Order
D2D	Device-To-Device
DB	Database
DED	Direct Energy Deposition Method
DOF	Degrees of Freedom
DOM	Document Object Model
EBS	Elastic Block Storage
EC2	Elastic Compute Cloud
EER	Enhanced Entity-Relationship
ERP	Enterprise Resource Planning
ETLA	The Research Institute of the Finnish Economy
ETO	Engineer-To-Order
FI	Future Internet
FTP	File Transfer Protocol
FTPS	File Transfer Protocol with SSL security
GCP	Google Cloud Platform
GMAW	Gas-Metal-Arc Welding
HATEOAS	Hypermedia As The Engine Of Application State
HMI	Human Machine Interface
HTML	Hypertext Markup Language
HTTP	Hypertext Transfer Protocol
HTTPS	Hypertext Transfer Protocol Secure
IaaS	Infrastructure as Service
IAM	Identity and Access Management
ICS	Information Centric Security

ICT	Information and Communications Technology
ID	Identification
IDE	Integrated Development Interface
IERC	IoT European Research Cluster
IGW	Internet Gateway
IIoT	Industrial Internet of Things
IO	Inputs/Outputs
IoE	Internet of Everything
IoT	Internet of Things
IP	Internet Protocol
IWS	Institute für Werkstoff- und Strahltechnik
JSON	Java Object Notation
LM	Laser Melting
LMD	Laser Metal Deposition
LS	Laser Sintering
MES	Manufacturing Execution System
MQTT	Message Queue Telemetry Transport
MTO	Make-To-Order
MTS	Make-To-Stock
MVaaS	Materialized View as Service
NASA	National Aeronautics and Space Administration
NAT	Network Address Translation
NIST	National Institute of Standards and Technology
NPM	Node Package Manager
ODB	On-board Diagnostics
OPC	OLE for Process Control
OPC UA	OLE for Process Control Unified Architecture
OS	Operating System
PaaS	Platform as Service
PLM	Product Lifecycle Management
RDS	Relational Database Service
REST	Representational State Transfer
RFC	Request for Comments
ROA	Resource-Oriented Architecture
RP	Rapid Prototyping
RS	RobotStudio
S3	Simple Storage Service
SaaS	Software as Service
SCADA	Supervisory Control and Data Acquisition
SDK	Software Development Kit
SFTP	SSH File Transfer Protocol
SLA	Service Level Agreement
SOA	Service Oriented Architecture
SOAP	Simple Object Access Protocol
SQL	Structured Query Language
SSD	Solid State Drive
SSL	Secure Sockets Layer
SSH	Secure Shell
STL	Stereolithography
TC	Trusted Computing

TCG	Trusted Computing Group
TCP	Tool Center Point
TLS	Transport Layer Security
TPU	Teach Pendant Unit
TUT	Tampere University of Technology
TVMM	Trusted Virtual Machine Monitors
UAS	User Authentication Service
UI	User Interface
UML	Unified Modeling Language
URI	Universal Resource Identifier
URL	Uniform Resource Locator
WADL	Web Application Description Language
VM	Virtual Machine
VPC	Virtual Private Cloud
VPN	Virtual Private Network
WRM	Wapice Remote Management
WSDL	Web Service Description Language
XaaS	Everything-as-service
XML	Extensible Markup language

1. INTRODUCTION

In past few decades world has been introduced with multible methods for web based application assemblies. One of these methods is called Service Oriented Architecture (SOA). The key feature using SOA lies in the architectural style by which the service consumer interacts with service provider offering the requested service. Usually this service is related to capalities of changing the state of service consumer [1]. Representational State Transfer (REST) is a arctitectursal model which was originally developed by Roy Fielding in his doctoral thesis. REST architecture was later on derived for RESTful services, which gives the quidelines for designing intercations between server based services. Thus, REST services can be used for building interfaces for SOA. [1; 2]

Cloud computing is another new service that is introduced in past decade. Cloud computing is sometimes refered as internet-based computing. However, the term cloud computing has been adapted for industry de-facto when referring to computing taken place either public (off premises), or private (on premises) servers. Another term is used for computing which includes both public and private servers compined together. This kind of architecture is called hybrid cloud. [3] Cloud computing term originates to occurences where software applications and other services have been moved to servers located in distant datacenters. Cloud computing additionally introduces three different architectural styles. Insfrastucture as Service (IaaS), Platform as Service (PaaS) and Software as Service (SaaS). Each of these are constructed on top of each other. However, depending on service provider these can be provided as individual services. [3] Now the first connection can be noted. Software as Service can be seen as a platform for providing Service-oriented architecture. SaaS provides a platform for service which gives the service consumer a new state or another information from where the consumer can continue. [4] One prominent method for connecting service provider and service consumer is a communication through RESTful based services [2]. Platform as Service is used when consumer gains access with cloud computing provider for using their platforms when deploying their own software. Depending on cloud provider some software languages might be supported and some not. Infrastructure as Service can be imagined as the lowest level of services. [3]

The most resent introduction to the list of technologies represented in last chapters is Internet of Things. Internet of Things has multible names according to the author referring to present technology. Thus, multitude of companies are trying to stand out when gaining more customers by making the variations over the title. Internet of Things

is often refered as abbreviation IoT and basically it means the technology where smart sensors are connected to the internet. These smart sensors can communicate with each other by Device to Device (D2D) method or deliver the sensor data to data servers. [5; 6] The reason for IoT been developed rapidly over the last few years, is the technology behind it. The costs for the electronics embedded in the smart sensors is lowered to level where deploying wast amount of these sensors is economically reasonably. Another reason is the expansion of the internet connection reaching almost every new device mounted in industry and in peoples homes. [5] IoT has provided also another advantage what comes to building user interfaces (UI); Dashboards. Many IoT service providers are marketing also their own version of IoT Dashboard, used for visualizating the data gathered from the smart sensors. The convenient use of these Dashboards can be extended to build data visualizations in reasonable time (with the help of cloud computing services).

As it is described, multible tools are exists just to be used. When accessing these tools consumers can exploit their own implementation for applications that they keep reasonable or profitable when pursuing new business models. The only complication is to notice the possibilities of each services and realize the potential which is waiting to be exposed. In past years increasing amount of companies are turning their interests into cloud computing and other off premises based technology. Duan et al heralds that cloud computing will have major role in companies development for upcoming years. [7]

1.1 Problem Definition

With the academic research, the data collected from the empirical studies is mostly recorded using spreadsheets or by some legacy based software's or some proprietary software provided by device manufacturer. Research data is not collected in structural form or either centralized manner. For some practical research and particularly in research where results are searched by means of repetition, a dilemma appears where more and more time is consumed in manual data collecting. The solution comes when the research data is recorded automatically and implemented data sensors monitor the environment. With these methods, the research can focus on the results, not recording the values correctly. Additional value comes later in the studies when all gathered data can be analysed to the core and effective phenomena's can be detailed. Another benefit can be noted if process can be controlled in real-time by implementing machine learning over the collected history data.

Similar requirements are occurring in private sector enterprises. Companies are searching for added value for their products and turning their vision for product service based business model. Gaining advantage with this new model, vast amount of data must be gathered for analysing and decision making purposes. Starting point with this new model is to gather the data efficiently. Discussion over the matter is represented in finish

news magazine Tekniikka ja Talous (Technology and Economy) [8]. According to the magazine from November 2016 future companies should be apple to transpose the data between their customers and suppliers more resiliently. Real-time monitoring of products and disposing of device data transfer boundaries are key figures for future growth. Lack of knowledge is one reason for resisting the open interfaces yet the change in the attitude of the company's personnel is another matter on the way of open data transferring. [8]

Both academic research and private sector goals could be achieved when sufficient amount of data can be collected from the processes, stored in structural form and represented to the user. After the initial phase where data collection is formalized it can be used for machine learning, controlling the process and for search of new business models. Reaching the goals is possible by novel cloud based solution where implementation is divided in two separate realization, backend and fronted. Backend acts as server collecting the data and providing it to frontend where data is visualized the data to user. Backend can be built on cloud services and fronted can be implemented with IoT Dashboard frameworks. When solution is designed in this manner, it aids researches to modify the data collection as research evolves. Similarly, private sector can more rapidly, with less human resources and less ICT (Information and Communications Technology) knowledge, search new business area and improve the existing ones. From these grounds, it is reasonable to study the possibilities of cloud computing acting together with IoT frameworks for finding the solution to problem set forth in above paragraphs. Solutions that serves both academic research and private sector companies. Finding the solution for presented issues with novel cloud computing paradigm is additionally reasonable after studying the future prospects. According to Frost & Sullivan [9] 40% of the global data will be stored in the cloud based platforms by the year 2020. Frost & Sullivan additional states in [10] that new cloud based services are on the rise

1.2 Work Description

Thesis makes theoretical search for cloud computing theory, cloud computing technology providers, Dashboard frameworks from the field of Internet of Things and interface methods for transferring data between different parties of assemblies. Thesis will also compare the features of the cloud providers and explain the differences in each technology. Thus, through the work a possible implementation prospects for small and medium sized manufacturing companies and academic research are kept in mind. Additionally study over the Additive Manufacturing method of Direct Energy Deposition is conducted. Comprehension of this method is essential for the reason that implementation is designed for this particular production process.

With the help of theoretical research, one of the multiple methods is selected to be the one used in implementation. The focus of the implementation is to build cloud based

environment for process data gathering and real time visualization in additive manufacturing research. When finished the researches can keep the focus on the research itself leaving the data recording and real time visualization of the system to the burden of the cloud framework.

1.3 Assumptions and limitations

At the initial stage of planning the thesis, some limitations and assumptions of the application level devices came clear. The environment providing the platform for implementing the designed solution is described in detail in the following chapters. In addition, the technology researched within the environment is also detailed. Both of these matters are essential for building the final data gathering solution for the reason that right variables are collected and substance data can be presented. For the readers of the thesis it comes easier to follow the coming chapters if some details and assumptions are described here at the introductory phase. These matters are:

- Universal robot acts as the manipulator in the environment
- There are no additional controllers, robot handles the controlling of the process
- Additive Manufacturing devices and tools have non open interface
- Lack of interfaces forces the robot to gather the main data
- Timestamping keeps on track when one device (robot) gathers the raw data
- Selected robot supports File Transfer Protocol, REST service, .NET solution
- Data gathering and visualization should be handled based on public cloud
- Cloud services should possess low learning curve
- Selected cloud service platform(s) should be ones relied for future existence

1.4 Methodology

Implementation of the environment is based for the theoretical background. Before the implementation may start the research over the following topics will be carried out.

- Familiarize the methods for additive manufacturing for understanding the requirements of the process
- Study over the theory behind the cloud computing technology
- Resolve the possible interface methods been used
- Research over the public vs. private cloud computing paradigm
- Take closer look over the IoT Dashboard solutions

Another half of the thesis is implementing the environment to the additive manufacturing environment. This part is constructed from the following parts.

- Configure and prepare a cloud computing framework for the implementation
- Handling of Real-time process monitoring
- Operations with Process data history
- Creating a Report for finished process

1.5 Thesis outline

This Thesis has five chapters. Chapter 1 covers the introduction for the subject including the problem definition, work description and description over the methodology. Within Chapter 2 the extensive study over the cloud based computing is been illustrated. Main task is to represent the factors from public and private cloud technology incorporated with the IoT Dashboard study. According chapter covers additionally familiarizing for the additive manufacturing and the search for the appropriate interface for data transferring. Chapter 3 takes closer look for the selected cloud computing technology and Dashboard solution been used in implementation. Second to last chapter, Chapter 5 has main task to cover the implementation part. This chapter describes first how the cloud framework is designed, configure and build. Second real-time process monitoring is detailed. Third part in the Chapter 5 illustrates how the process data is gathered. After gathering, data is passed to cloud service where it is manipulated and finally visualized for the user through Dashboard solution. Fourth part of the according chapter is to portray how the report creation of the process is carried out. Chapter 6 concludes the thesis giving the analysis over the work and gives proposals for the future development of the system.

2. THEORETICAL BACKROUND

This chapter takes an extensive coverage over the theoretical issues accessed within thesis. For the thesis becoming a reality, an implementation subject had to be found. Through the groundbreaking work with the additive manufacturing field at TUT (Tampere University of Technology) Laser Application Laboratory an implementation environment was discovered from in questioned field of manufacturing. Thus, the first Chapter is dedicated to additive manufacturing technology and specific variation of according field, Directed Energy Deposition (DED). Chapter continues by introducing cloud computing which is the basic technology used in implementation part. The major issues in cloud computing, security and reliability are covered with survey over the articles and reports. Another foundation in the implementation comes though Internet of Things, and more over the IoT Dashboards. Thus, chapter takes closer look for current state in IoT technology. Later on the chapter, covers data transferring methods available in the implementation part. Latter sections are dedicated to theoretical work on studying cloud computing providers and IoT dashboard technologies.

2.1 Additive Manufacturing

Additive Manufacturing has the origins from the 1980's after which the other technologies, computer-aided design (CAD), computer-aided manufacturing (CAM) and computer numerical control (CNC) reached the maturity level for producing three dimensional objects and so worth making the questioned production technology possible. [11] Another significant impact on the rise of AM technology was STL (Stereolithography or Standard Tessellation Language) file format developed by 3D Systems Inc. Inside the CAD file object shape is stored in continuous geometry. Converting a CAD file to STL file format translates this continuous geometry into a header and small triangles added with the normal vector of these triangles. When processing the STL file for AM production, the file is cut in slices each sliced layer holding the points and information of that specific layer. [11] This information can be inserted, into G-code file. AM devices can then read a G-code file format and manufacture the artifact accordingly. Additive Manufacturing follows different discipline compared for conventional manufacturing where material is being removed from the blank. Like described in [12] Additive Manufacturing process artifact is formed layer by layer from feedstock normally consisting of wire or powder.

Technology as we know it today was not always called Additive Manufacturing. In the 1980's Rapid Prototyping (RP) was the term for same ideology. Initial drive for creating

such manufacturing method was the urge for creating prototypes over the artifacts, thus portraying what engineers have in their mind. Formal RP technologies enabled furthermore a reduction of time and cost moreover making possible of creating pieces impossible to machine. Novel technologies in the AM have made it possible to manufacture finished product straight out from the AM device. AM processes are evolving to the level where no polishing, machining or abrasive finishing are needed. All the possible solutions for AM processes are yet to be found. Some of the use case examples at the moment are architectural design of buildings and structures, medical applications via biomedical materials and 3D scanning processes, manufacturing of lightweight machines from exceptional materials or by structural concepts. Artists have their own intentions for making novel objects. One user group of the AM processes are the hobbyist making extraordinary artifacts and repairing household products via printed spare parts. [11]

AM processes can be classified into three main categories representing the material used in the process; liquid based, solid based and powder based solutions [11]. However, these categorizations are quite straightforward and multiple other categories can be considered. Alternative concept for categorizing the AM methods are through the energy source or via the method of how materials are joined [13]. Categorization is furthermore possible by the material being used; plastic, metal or ceramics [13].

Subcategorization of Additive Manufacturing through the method of feeding the energy into the process leads to technology called Direct Energy Deposition [14]. DED is a method commonly used for adding additional material on already existing part or repairing damaged artifacts. DED solution consists of manipulator having multiple degrees of freedom, practice which enables the addition of material in any part of the artifact. Manipulator holds a nozzle (tool) from where the material is deposit on the objects surface. Near the surface material is first melted and on the surface the deposited material is finally solidified. DED method can be used for ceramics or polymers, yet the most common solutions are built for metals. Deposit material can be inserted either with wire or by powder and the melting can be arranged either with laser or electron beam. [14] DED method is the one used in the application environment of the thesis. Environment in which the data gathering is implemented.

2.1.1 Path manipulation

For making both research and solution testing with Additive Manufacturing and its subclass Direct Energy Deposition, there is a need for versatile environment. One part of this environment is the manipulation method for the different tools (nozzles) used. When searching a commercial solution first intuitive manipulator is an industrial robot. Industrial robot with 6 degrees of freedom (DOF) has the asset of reaching all the points in the toroidal working area. However, the challenge rises if more advanced manufac-

turing in different poses need to be conducted. Cladding of a rod is one example. Process can only be handled when rod is positioned vertically and is rotated simultaneously during the cladding. Solution for this and other similar problems is additional manipulator called positioner. Positioner in this case is a 2-axis device capable of rotating its table and horizontal axis. According devises from the application environment are illustrated in the Figure 1.

Figure 1. *ABB IRB 4600 robot (left) and ABB IRBP A-750 positioner, adapted from [15; 16]*

Another issue for the additive manufacturing solutions is the accuracy of the manipulator. There could be a significant difference between the position in the virtual controller model and the actual robot. For the robot (ABB IRB 4600) of thesis implementation part there is a concept called Absolute Accuracy [17]. Absolute Accuracy compensates the mechanical properties and the deviation of the axes due to the payload. Through the implementation of the questioned approach robot can maintain accuracy of 0.5mm inside the working area. Usually industrial robots work inside 8-15mm accuracy. Technology for gaining the 0.5mm accuracy lies in the proprietary algorithms inside the robot controller. Because solution for the problem is non-linear and complex, ABB has resolved the issue with a position compensation inside the controller. Robot adopts the kinematics from the generic library of the particular robot model and the actual position is reached using compensation parameters collected with 3D measurement system. [17]

2.1.2 Cold Metal Transfer method

One possible Direct Energy Deposition method is an approach of Cold Metal Transfer (CMT). CMT is a welding technology developed by Fronius International GmbH [13]. Before describing the technology further, few words over the conventional welding process. Fusion welding is a concept where heat is applied to the welding groove to create liquid weld pool [13]. Afterwards the weld pool solidifies and creates strong and per-

manent joint. Source of the heat could be a flame, laser, electron beam or, the most popular one, an electric arc. With the welding process, there is also a possibility to add a filler metal into the weld pool and so worth fill the gaps of the object. The most employed fusion welding method is Gas-Metal-Arc Welding known as GMAW. In this method filler, metal acts as the electrode for the electric arc meanwhile filling the weld groove. Electric arc is formed between the weld groove and tip of the filler material. Electric arc melts the tip of the filler and creates a common weld pool. Atmospheric protection is performed with shielding gas. Reason for the favor of the welding and more over GMAW comes through the fact that each type of steel, aluminum, copper and nickel alloy could be used as the filler wire. By using the weld torch and manipulator for overlaying the successive weld seams the technology can be used for AM processes as well. Welding process is, in addition, an easy task to be automated. [13]

Austria based company Fronius had an idea of developing a GMAW solution for welding steel together with aluminum. The criteria for accomplishing questioned task is to avoid the mixing of these two materials. In other words, steel has to remain solid while aluminum molts meaning that process has to work on quite modest energy level. Fronius has resolved this matter with high frequency (130 Hz) forward-retract movement of the filler wire during the welding process. [13] According device has both digital controlled wire feed and digital detection of electric arc short circuit. When the short circuit is initialized, the retraction of the filler takes place meanwhile the arc is extinguished. Consequence of this method is the release of the molten droplet form the filler material (see Figure 2 for details). Thus, thermal effect is reduced causing the term Cold Metal Transfer or ensemble CMT-GMAW. [18]

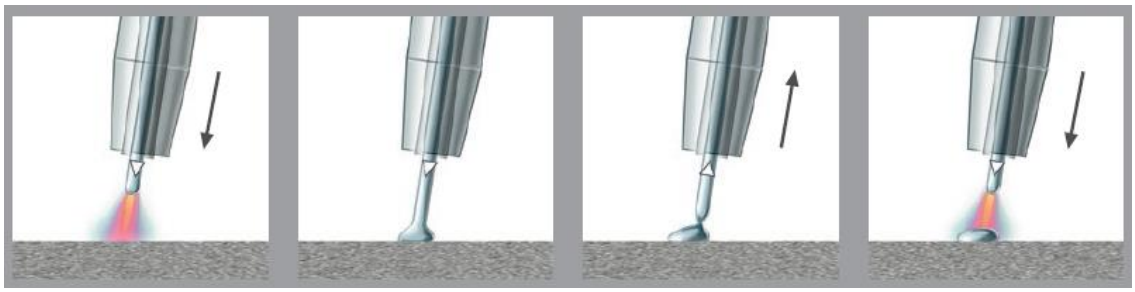


Figure 2. *Cold Metal Transfer process [19].*

Vast range of metal materials and alloys can be used with CMT and the process itself reduces the spatters often present with the conventional GMAW process. Minimum thickness of the seam created with the CMT process fluctuates by the diameter of the filler metal. If wider seam is requested the action of weaving with the manipulator can be initialized. Reduction of thermal effect and properties mentioned has raised the opportunity to use the CMT process for Additive Manufacturing and for DED solutions. [13; 18]

2.1.3 LASER aided Additive Manufacturing

Studying laser technology implemented in the field of AM, three different methodologies are addressed. Laser Sintering (LS), Laser Melting (LM) and Laser Metal Deposition (LMD) are the current most versatile technologies used. [12] However, regardless of the versatility of the methods, each of them are a composition of complex chemical metallurgical and non-equilibrium processes where heat and material transfer plays significant role. In novel laser additive manufacturing processes, the substance can be delivered either in the form of powder or filler wire. Process itself is highly dependent of the materials chemical constituents, substance particle size and shape, packing density and the flow ability of the powder (when powder is accessed as the substance). Equivalent importance in LS, LM and LMD comes through the process values of laser power, laser spot size, speed of the scanning and type of the laser. [12]

Laser Sintering is one alternative for laser based AM processes. In LS manipulator, levels powder substance layer by layer and sintering is conducted with laser energy. Atmospheric protection of the powder and preheating of the build platform has a significant role for contriving with this method. Selection of the laser technology (fiber laser, disc laser, Nd:YAG or CO_2) is important considering the fact that different substance materials absorb different wavelength of light in divergent ways. In addition, the metallurgical mechanism in the process is determined with laser energy density. Sintering time varies from 0.5 to 25 ms which causes the melting/solidification reaction. [12]

Laser Sintering is not the solution when demands are considering the fully dense components with no time consuming post processing phases. To meet these requirements Laser Melting technology is developed. Application solution for LM process shares the similar devices with LS technique yet the difference comes from the complete melting/solidification reaction when compared with LS. LM method is enabled by the enhanced properties of the laser. Key improvements are higher laser power, smaller focused spot size and superior beam quality. All this leads to advanced microstructural and mechanical properties when compared with aged LS solutions. However, LM process, occupies complications. During the process, the substance lies in the molten pool state, which can come instable and ruin the artifact. Constructed artifact sustains high stress consequent from the shrinkage during the transformation from liquid pool to solid material. Problem that could cause the distortion or delamination of the finished product. [12]

Final conceivable method for using laser in additive manufacturing is Laser Metal Deposition method occasionally referred as Directed Energy Deposition [13]. Some of the principles from LS/LM methods are adopted yet the compelling contrast comes from the powder feeding technology. In LMD, powder is fed through specially designed nozzle system where gas driven powder feeder delivers the substance from center of the

nozzle. From the same nozzle laser beam is injected to the work piece by focusing the beam close to the surface. Focused beam melts the powder and the substance is solidified on the work piece. Controlling the z-axis movement, the layer can be altered and by controlling the x and y-axis arbitrary forms can be manufactured. In composition three dimensional artifact is produced. By implementing LMD (DED) method it is possible to repair, coat and build artifacts with complex geometries. [12] Coating gives additional value where artifact with lower hardness or corrosion resistance is enhanced with the layer of superior material [20]. LMD (DED) is highly versatile process for studying and manufacturing various artifacts. Different laser AM techniques using the powder or wire as the substance are aggregated in the Figure 3.

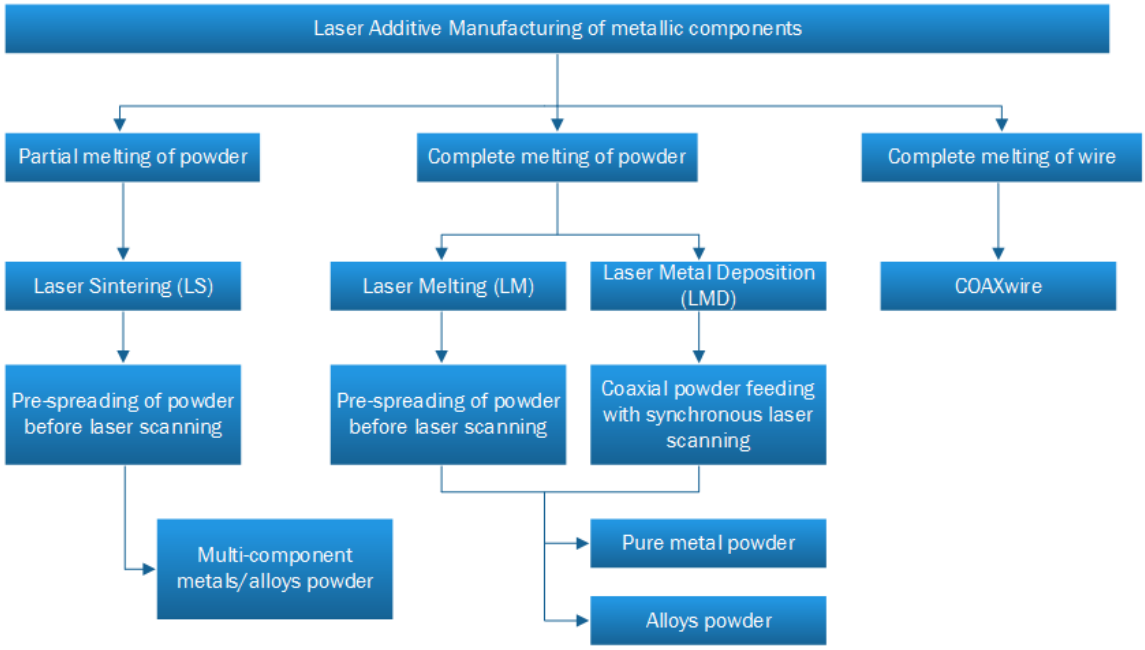


Figure 3. *Laser Additive Manufacturing processes classification, modified from [12; 21].*

Powder based systems enable the forming of thin structures at narrow targets. If generalized, only the laser beam and sustainable amount of powder are required to make the structure. Powder techniques in addition does not require great precision and timing at the points where beam is enabled and disabled. Simplified ideology is that only adequate amount of powder is present. The disadvantage of powder system is the loss of the substance falling from the target, not been melted. Powders furthermore holds a risk for human operators where the substance can find its way inside the human body by breathe if sufficient protective gear is not used.

German based research organization Fraunhofer and its subdivision Fraunhofer Institute of Material and Beam Technology IWS (Institute für Werkstoff- und Strahltechnik) Dresden [22] has developed a concept and device called COAXwire [21]. COAXwire stands for Coaxial laser wire cladding head [21], a laser head which can use wire as the

filler element. COAXwire is designed to afford welding process with omni-directional performance. Fundamental of the head is based on the beam splitter which divides the collimated laser beam in three separated beams travelling at the outer shell of the device. Three beams appear from three optic nozzles at the bottom of the device and one unified laser beam is constructed at the focal point. Optical elements of the COAXwire are constructed in a way that beam focuses exactly at the center axis of the filler. Filler runs in the centerline of the device. This causes an action where filler is injected precisely in the center of the laser created molten pool. Causing the advantage where all welding directions and poses are conceivable without the interference from gravity. [21] All this has being enabled by the development of digital technology at the stage where start and stop sequences of the beam and filler wire feed can be handled with sufficient precision and timing. Fraunhofer IWS COAXwire is one device in the application environment of the practical part in the thesis. COAXwire moreover composes a third line of technology in Figure 3.

2.2 Cloud computing

The purpose of next paragraphs is to take a theoretical view over the cloud computing. Cloud computing is fairly a new ideology over the issue of how the server based computation and services could be organized. For this reason, terminology and basic functions are settling at the moment. One of the newest cloud based services are Internet of Things (IoT) solutions. These solutions provide interfaces for connecting devices to internet for the purpose of monitoring and controlling processes.

Section 2.2.1 describes the theoretical concept of the cloud based computing. Section portrays both the historical background of cloud computing as its current state. Following section, Section 2.2.2 concentrates to security issues and reliably of the cloud solutions. Both of these issues are most important then building a cloud based solution and more over when shifting businesses to be solely located on the cloud. Failing with reliability and security might cause the company's core business to fail with catastrophic consequences. Last section, Section 2.2.3, takes closer look into Internet of Things solutions, technology itself and current state of the applications. Cloud computing in general, furthermore with IoT technology, lies in quite introductory state and both the technology and the platform providers are basing their grounds.

2.2.1 Cloud computing concept

Cloud computing can be comprehended as all the services which takes place out of entities own premises and are accessed over the internet [23]. In addition National Institute of Standards and Technology (NIST) defines the cloud computing to be a model in which an universal access is based on the commission and where computing resources can be configured and changed, including storages, servers, networks, services and ap-

plications [24]. Premises in here can be sorted out to mean individual persons' homes or facilities for some particular company or business. To be more precise cloud computing covers all the activities that takes place over the internet which incorporates the use of devices, services or, more anonymously said the use of resources located on providers web servers [25]. Reference to the modern cloud computing can be found from the history. Cloud computing is a combination over the grid computing, parallel computing and distributed computing [26]. The basis of the thesis is to concentrate on the cloud computing resources used by businesses but for the wider audience it can be mentioned that most of ordinary people use cloud based solutions in every day basis. For contacting their friends and family or when using web based banking solution. Most of the people never realize that the usage of electronic mail is also a use of cloud computing [25].

Cloud computing is based on large data centers which are maintained by the cloud provider [27]. Within these data centers, vast amounts of physical resources are running simultaneously. These physical resources are then applied by multiple virtual machines (VM's). Each of these VM's can represent one ecosystem whereas this ecosystem are the virtualized locations of cloud consumers. Cloud consumer can have one or many of these VM's and so forth use the cloud as on-demand. Scalability, on-demand resources, resilient computing, recovering from disaster and extensively high performance are the main features for justifying the use of cloud based computing. Major players on cloud computing field are also providing payment methods where you only pay for what you use. [27; 28] For small and medium sized enterprises, this creates a significant asset. Costs for using the cloud are substantially lower when compared to the technology where computing power is maintained on private servers [25; 29; 30]. All this also has another side; cloud data centers are remarkably sophisticated infrastructures. Orchestration of cloud resources makes the cloud solutions both, vulnerable for security and reliability, yet accessing cloud resources require additional expertise through the lack of standardized interfaces [27; 28; 31].

Cloud computing theory holds three different models for describing the level of services whom cloud providers are offering. Frost & Sullivan addresses these level as the legacy levels, for the reason that new business insights are on the horizon [10]. The lowest level of basic service is Infrastructure as Service (IaaS). IaaS is a service where service provider offers only the physical resources accessed by the consumer. Consumer must deploy their own operating system (OS), data storage methods, software's and network connections. [3] Platform as Service (PaaS) represents the middle level out of these three service stages. PaaS is a realization of physical resources been submitted for one virtual machine. Used resources can be distributed over various data centers; however, virtual machine acts as one frontend entity. Consumers exploit this platform as one environment for deploying their own services and computational applications. Service consumer can maintain virtual machine(s) via internet browser based portal and through

this portal changes can be made for platform within the limits of service provider. [3; 26] On the highest level locates the Software as Service (SaaS). Method and ideology for using SaaS diverts totally from two mentioned ones. The principle is that SaaS is accessed on-demand bases and through any device, that has an access to internet. Service consumers possess only minor possibilities to customize the service. SaaS acts as an individual entity that is used to change the state of the consumer or provide a new thread for the consumer to continue its actions. [3; 26]

These three layers forms the basis for cloud technology, although multiple variations exist. Singh et al presents a new form of PaaS named Plat Serve derived from Platform as PaaS [26]. Plat Serve stands for the paradigm where all the operating systems are installed on the central server and user only picks up the one, which is required at a time. This gives the advantage over traditional problems for operating system updates. All the updates are always activated through this one master OS, gaining the user an access to the latest features. [26] Additionally, variations of different services can be combined and illustrated with equal basis. These variations include, among others, Communications as Service (CaaS) [3] and Materialized View as Service (MVaaS) [32]. Consulting company Frost & Sullivan introduces a model of Everything-as-service (XaaS) in their report [10] of new business opportunities in cloud services. Frost & Sullivan states that new services will emerge and more increasing amount of services will be offered as cloud based.

Dialogic portrays four different cloud location models in their Whitepaper [3]. These four models are also addressed by Duan et al in their article for a Construction Method and Data Migration Strategy for Hybrid Cloud Storage [7]. Private cloud is addressed as cloud located entirely within locations firewall. Private cloud can be maintained either the operator itself or some third –party operator. [3; 7] Private cloud has the advantages for storing vast amount of data with high reliability in terms of availability [7]. Community cloud portrays a cloud deployment model where multiple consumers share the same cloud infrastructure [3]. Usually these consumers have similar requirements for the cloud and so forth the usage of same deployment is conceivable. Consumers may also hold a demand for allocating rather modest amount of financial resources for deploying their function at cloud. In these circumstances, a public cloud comes in question. Public clouds are commercial versions of cloud based computing and can be accessed with rather modest payments for the provider. Financial model in these clouds are based on pay-as-you go type of invoicing. Hybrid cloud is a composition over two or more of these three other explained cloud types. [3; 7] When deploying a hybrid cloud two main issues should be covered. Usage of two different cloud types should be invisible to the end user, and at the same time implementation should hide the complexity of the structure behind the multiple cloud system. Gaining these two aspect at the same time is much more challenging than commissioning a one model based cloud system. [7]

2.2.2 Security, privacy and reliability

Each entity mentioned in the header of this section is one of the most important issues related to cloud computing. According to survey conducted by Benslimane et al, slightly less than 74 percent of the total 203 papers them were able to discover, related to security issues concerning cloud computing [33]. One of the issues in inflation of cloud based systems in business solutions are related to security and privacy of the data stored in cloud [34].

Multiple aspects over the security matter can be formed. Security and privacy can be seen as separate elements hovering over or encapsulating the actual cloud based system. Such as Virtual Private Network (VPN) connection shielding the communication between cloud and the consumer. Further, these matters can be comprehended to be incorporated in each actions performed with cloud based system, thus creating a fabric where data and security are co-existed. [34; 35] This leads to new way of understanding the development of software's that are deployed in cloud based ecosystems. Software development process has its standard manner how different stages are handled. In legacy systems software developing consist of planning, modeling, construction, communication, testing, deployment and maintenance. [35] However with cloud based systems software designers should keep in mind that the data is transferred to cloud ecosystem with public connections and internet backbone switches. Thus, these mediums need to be secure as well. Consumers of cloud based systems should also keep in their mind that security is not only a responsibility of the cloud service provider. Consumer itself should be aware and handle the security from their part. [34] One of the first solutions for security issues is that consumers turn their look for is the Service Level Agreement (SLA). SLA is a document where service provider gives their promise for, security and performance of the service according to the contract in which the consumer has agreed. Some of these SLA agreements could also include additional information such as data location and its auditability for service provider. [33] In principle, SLA is based on the trust between service provider and service consumer.

Narula et al has reviewed the matter of trust through a pair of concept covered in their study over cloud computing security [34]. Trusted Computing (TC) and Information Centric Security (ICS) [34]. Cloud service providers are constantly enhancing the security related to cloud computing. In many cases, the eventual trust can be accomplished by introducing, a third party for authenticating both cloud consumer and cloud provider. Narula et al identifies these actions as remote server attestation. Mainly the idea in TC is an encryption, which is conducted for the information, and the decryption key is provided for trusted program. Operation is handled via third party hardware chipset installed on computers. [34] Narula, Jain and Prachi additionally describes the Trust Computing Platform [34]. TCP was originally a title of the group providing the third party security. At the present state the platform owns the title of the technology been used and the or-

ganization has been renamed as Trusted Computing Group (TCG) [36]. TCP is based on two services. Other one is authenticated boot and other one is encryption. These actions are conducted via Trusted Virtual Machine Monitors (TVMM) and Trusted coordinator. TVMM acts as hosting the customers' virtual machines and Trusted Coordinator runs these VM's in secured location shielded by security perimeter. Only the combination fulfilling both of these requirements are then relayed to be trusted ones. [34]

ICS is understood as security of information over the security of medium where information is moved from location to location. ICS is based on encryption where only the author with legitimate decryption key can access the information. Dispute in these actions comes through the practice where in general both information and data is processed in the cloud without any encryption.

Confidentially, Integrity and Availability commonly labeled as CIA are extensively present at every article concerning cloud services [34]. Confidentially means that data and information held in cloud ecosystem should be encrypted for any unauthorized access. Only the entities having the access can reach the data. Integrity concerns both the data and information. These should not be modified by any unauthorized personnel neither any process nor entity. Especially the information inside the cloud should remain consistent. Reliability of accessing the data or computing resources should be consistent through all the timeline consumers holds the access to the kept resources. Service provider is responsible for providing backup methods and concurrent VM's to provide the consumer with Availability at any time. [35]

Trust was identified as agreement between two parties of cloud operators verified with third party player. Similar issue can be noted for integrity of the data and information within cloud systems. Cloud provider cannot acknowledge if the information was tempered with any hacker while it travelled in the interchange medium. Further cloud consumer should not have to care about the integrity once the information has left its source. Similar to Narula et al concept, Madhubala introduces in article over the security in cloud computing a third party member for watching the integrity and providing transparent actions for cloud consumer. [35]

In the implementation, the process data is delivered with File Transfer Protocol (FTP) although real-time monitoring is handled with Representational State Transfer based on Hypertext Transfer Protocol (HTTP) and Hypertext Transfer Protocol Secure (HTTPS) messages. Process data moves in text file where all the parameters are only numbers for any outside viewer. Only the source and the destination knows the sequence and relation of these data records. Complementary method is noted in Kaur et al article where they give a solution example by means of Image Steganography [37]. With this technology, a sensitive information is encrypted inside an image and decryption is conducted with Pixel Key Pattern where edge detection excavates the encrypted information.

After the data has left the premises of its original owner it is stored in large data centers beside several other cloud user's data. This causes an issue over the privacy of the data. Cloud service providers might have disclaimers in their SLA's for storing the data and its location. However, the owner has partially lost the control over the data stored in the cloud. Storing the data in large data centers possesses also a thread with itself. Most cloud providers compose a multiple copies of the data for availability issues. Data should always be accessible even with one data center disconnected from the internet. For this actions data owner (consumer) is not anymore aware of who and from where has the access to the data copied by automatized applications. Privacy issues raises likewise with sensitive data storing. For instance, names, phone numbers, addresses both personal and internet protocol (IP), medical history or criminal investigation evidence. [24]

Reliability in cloud computing intertextures with availability of data and information. Both Li et al [27] and Zhang et al [38] introduces a different perspective in cloud reliability in their studies. Zhang et al takes an aspect of reliability to represent it as the reliability in response time of cloud actions. They also state over the energy consumption paradigm in cloud computing in which total energy applied by data centers is increasing by 12% each in year in US. Against these basis Zhang et al introduces and implements a queueing algorithm, which improves the response reliability time and meanwhile lowers the energy consumption [38]. Li et al study in the same matter has a different perceptive. In their study, the concept is based on reliability of the different physical servers running in data centers, serving as platform for VM's used by cloud consumers. If a physical server should fail, the VM's fail at the same time and the consumers cannot receive the reliable service. Conclusion for their study represents a state-space model with a combination of fault tree. With these tools, first making a state-space model of the physical server and then calculating the probability of each state within the fault tree can clarify a reliability. Outcome can give the cloud provider a means to evaluate the amount of physical servers running consumers VM's. [27]

Trust, as it is described in this section is otherwise a considerable matter with a technology called IoT for connecting devices to internet. Next section covers the usage and potential of IoT for possible technology bringing benefit in future manufacturing

2.2.3 Industrial Internet of Things

Megatrend is a global term, yet in the field of technology, it can be comprehended as indicating a major long-term change that takes place with some specific field without anyone actual effecting the direction of the change. Internet of Things, usually referred as IoT from the initials, is one of current megatrends. [5] Basics for IoT is connecting smart devices into internet but it also possesses frameworks (Dashboards) which brings cloud based services to the level where deploying solutions at incorporated level comes

more practical and convenient. Further, on within thesis cloud computing is conceptualized as backend solution and IoT Dashboard as frontend solution to be accessible for the users. This section of the thesis holds extensive look for current state in the field of IoT and its adjacent solutions.

IoT is a term that is profiled to cover products and services for private consumers rather than for the industrial sector. IoT considers smart devices connected into internet through which users can monitor their everyday life, health and surroundings. As for industrial IoT equivalence there is a concept called Industrial Internet of Things, regularly spoken of IIoT. [5] Designing of IoT devices and services starts from ground level. From the question of how consumers' needs can be more efficiently fulfilled and what are the prospects for making more economical and high-speed sensors. These sensors are used for relaying information and so worth producing more value to end user. When conversing over IIoT the methodology is different. IIoT glimpses the needs for specific industrial sector from bird's eye view trying to gain more efficient overall process and optimizing the needs for entire corporation. Lower level requirements for producing the devices and services are detailed after the higher-level concept is clarified. [5]

The rise of IoT and IIoT can be observed to be the consequence of internet connection taking more coverage and electronics coming smaller in size to be fitted in every device. Yet, the most important single matter enabling the rise of IoT is cloud computing platforms making the usage of the gathered data more convenient. General Electric points IIoT to consist of three main topics, smart devices, advanced analytics and humans working [5]. IIoT is additionally referred as the third industrial revolution [5]. The Research Institute of the Finnish Economy (ETLA) has released a report over the IIoT prospects in Finnish industry and manufacturing [5]. From this report it can be noted that possibilities for economic future growth with the help of IIoT are extensive. Similar studies are taking place all over the world and as an example; German has started their spearhead initiative Industrie 4.0 in year 2013. Questioned initiative is concentrating for flexible manufacturing systems, individualized manufacturing and integrations of both subcontractors and customers. With the addition of pursuing the additional value for the products and composing hybrid commodities. [5] Name of their initiative is indicating the fourth industrial revolution which should be, in their opinion, consisting IoT and Cyber-Physical Systems where cyber networks and physical world are tied together [5].

Multiple industries have turned their vision from producing basic commodities for providing life-cycle management. IIoT enables expedited growth for these services through data gathering from both devices and products. By analyzing this data, companies can provide predictive measures. ETLA's report states that business has three opportunities. First, they can emphasis their commercial function, method known as evolution. Second, they can pursuit new businesses, method known as revolution. Third, companies can reach for additional value inserted inside their products. [5]

Service business model can be twisted to be incorporated with the manufacturing industry itself, within the company. Before taking closer look for this aspect, it has to be noted that in legacy systems industrial sector has largely relied on the intranet connections. By introducing internet starting from the Enterprise Resource Planning (ERP) to Manufacturing Execution System (MES) to SCADA (Supervisory Control and Data Acquisition) system and finally on the shop floor, the methodology of services can be used and develop to be an internal service for providing more efficient manufacturing methods. [39] Perceiving the service based model as an internal function between different manufacturing devices, cells and subsidiaries, companies can reduce the input assigned for internal information systems and release these resources for another use. Although The Research Institute of the Finnish Economy points out that modern cloud computing and IIoT or IoT platforms require high-ended expertise, it seems that in future this paradigm is changing [5]. Platform providers are constantly raising the abstraction level of their services making the learning curve more low gradient. Companies can additionally apply public cloud computing model as the backend for the production devices. Described method opens a new set of services able to be provided for company's customers by the means of more precise and detailed production data. Data, which is collected in, formalized manner, not by users, so any human errors are eliminated. Now internal processes itself comes money worth information. [5] Described paradigm is portrayed in the Figure 4.

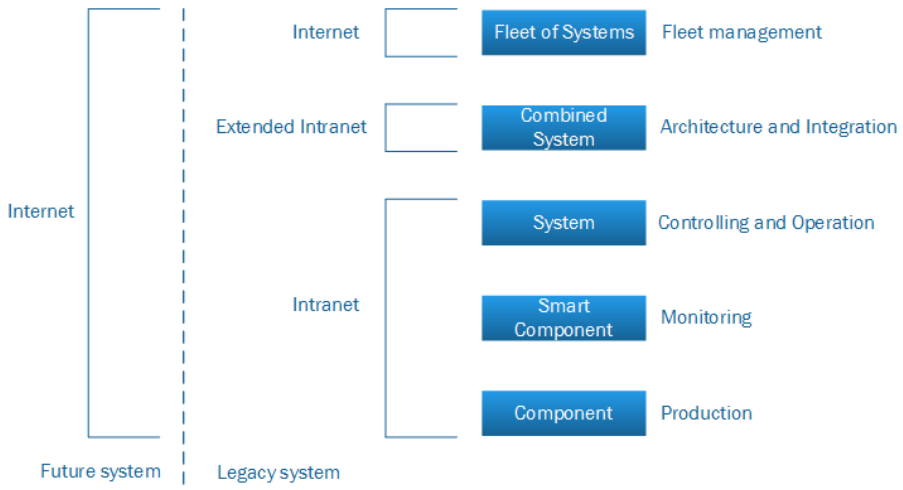


Figure 4. *Internet can be deployed inside the factory, adopted from [5].*

Previous chapter describes the concept of service-based architecture within company's internal structures as well in company-customer boundaries. The potential in Industrial Internet concerns also the future manufacturing performance. Soldatos et al [39] has taken an extensive view over the matter. In their study, they point out four different aspects for future manufacturing trends. First, is the emerging need for shifting from capacity ideology to capability ideology. In this concept, companies are changing their

manufacturing for more responsive and flexible methods when responding on market demands. Second, they point out the support for new production models. Factories are shifting from make-to-stock (MTS) methodology to make-to-order (MTO) and more over configure-to-order (CTO) and engineer-to-order (ETO) methodologies. These new models are replying for the growing need of mass customization. Third trend, which they propose in their study, is a movement towards proximity sourcing and proximity production. In this concept, factories, material suppliers, distributors, retailers and sub-contractors work together in tight loop for making modular products in common platforms. When applying mentioned manner stakeholders are able to perform the final customization at a location and provide highly custom-built solutions. Fourth point in Soldatos et al study is the change of work force commitment. Work forces are shifting from manual labor to more upper level when concerning factory workload.

Future manufacturing is highly leaning against Future Internet (FI). Manufacturing will change to be consisting of Industrial Internet, Internet of Things and cyber-physical Systems (CPS) where physical devices and cyber technologies act together. [39] These goals can be gained by the implementation virtual manufacturing applications with the actual factory automation, both incorporated by means of IoT as described in Soldatos et al work [39]. Future virtual manufacturing environments will be built in a manner where whole chain from material supplier to customer can be covered. Likewise, manufacturing methods are under rapid changes. Additive and Digital manufacturing will revolutionize the technologies used in legacy systems. [39]

Singularity point of these modern manufacturing processes are yet to come and several reasons for this are pointed out. Industrial sector is rather conservative when changing their methods and no actual operational and extensive pilot cells exists at the moment. Even though service oriented architecture paradigm appears very prominent. Another reason comes from the migration path and its lack of smoothness. Factories cannot change overnight and no clear methods is found for making the merging of legacy and future systems easy. Otherwise, standards for future solutions and techniques are still under specification. [39] Numerous initiatives across the Europe are taking a participation for intensive research in the field of digitalization. IERC (IoT European Research Cluster) is a consortium for hosting several topics for future manufacturing. Another union is AIOTI (Alliance for IoT Innovation) which has pointed a working group for studying future smart manufacturing. Numerous other projects within EU's FP7 and H2020 programs are also concentrating for future digital manufacturing by means of IoT. [39] In the practical part of the thesis one possible solution for future data gathering and visualization is presented through multiple technologies of SOA, REST services, IIoT, cloud computing and legacy system of FTP.

Interfaces between different devices plays a major role in digitalization and in future manufacturing, controlling and quality assurance. The concept of interface can be un-

derstood either located on the physical layer or at the API layer (Application Protocol Interface). There is an endeavor for unified interface connection between ERP, MES, PLM (Product Lifecycle Management), SCADA and shop floor controlling systems. [5] In legacy interfaces and systems there has been a concept called silos. Device manufactures have been using their proprietary interfaces thus causing a vendor lock-in situations. In past years the usage of these proprietary interfaces are diminishing. Mentioned silos could have been industry specific or even a specific within some industry sub level. This has caused a situation where it has been cumbersome to connect multiple devices together. According to new principles in IIoT these interfaces should be open and interoperability should be convenient. [5] Devices should also be able to interact with each other using D2D (Device-to-Device) communications methods. By doing so devices can form a mesh like structure for transferring information. D2D communication can be conducted with wired or wireless communication approach. [6] In the thesis application, Cold Metal Transfer (CMT) device is one example of silo principle. CMT device hold's EthernetIP bus connection. Connection is part of Common Industrial Protocol (CIP), although Fronius has closed the interface for any user outside their cooperation partners. [40] Described method can be compared with ABB robot controller, which holds REST interface for reading and writing the robot variables [41].

Designing an open and comprehensive interface for IIoT comes cumbersome through the variety of devices communication protocols and data formats, which are usually heterogeneity [42]. Domenech et al [43] introduces a concept named Smart Gateway placed between IIoT devices and cloud platform. Smart Gateway acts as proxy and is responsible for altering the device proprietary communication protocol making the device characteristics available through RESTful web service. Second, this proxy layer sends the data to cloud service and third it enables the remote monitoring of the devices. [43] Their solution consists of four components. Device Driver (first layer) abstracts the device communication protocol and transposes the data for Interpreter (second layer, responsible for constantly updating the virtual model of the device. On the third layer, RESTful Web Services provides resource of the device for clients to be accessible through HTTP messages. Fourth layer executes the RESTful Web Client, which sends the gathered data to cloud platform for future analyzing, visualization or other availability. In their model, data is transferred with JSON (Java Object Notation) instead of XML (Extensible Markup language). JSON was selected for its need of less computational resources when processing the data and for the nature of JSON being less verbose. [43]

Emeakaroha et al represent a contrary method for Domenech et al search, called Generic Cloud Interface [42]. On top of this Generic Cloud Interface, they propose a Server Middleware making the connections to actual devices through device's heterogonous interfaces. Their study additionally introduces a direct link from Generic Cloud Interface to devices when controlling of actuators are in question. Generic Cloud Interface

composes from three layers. First layer acts as security authenticator enabling the authentication methods for the connected devices and thus assuring the privacy of the data. Second layer plays the role of formatting the data. According layer gathers all the data and transforms it to the platform-neutral format. Final layer is called Communication Mechanism. This mechanism has two different models. One for the more generic communication via HTTP messages used in large scale applications which support according technique and one for the D2D communication handled with method called message bus. Message bus is composed of three individual agents, producer, messaging infrastructure and consumer. In the operation, producer and the consumer has no need to know about each other's capabilities and interfaces. [42]

Both Domenech et al and Emeakaroh et al proposals are quite far similar. Domenech et al do not have the feature for D2D communication and they do not take a stand in the security issues, which are quite profound when acting with IIoT technology. User acting with IIoT and cloud computing are concerned about their data privacy as described formerly in this thesis. With IIoT technology user should be more awake with the security issues [42]. Moreover, understand a concept of trust when referring IIoT or consumer counterpart IoT.

Concept of trust is used when dealing with human beings, yet the rise of the devices being connected to internet the mentioned trust is altered to comprehend the matters between entities of any kind [44]. Tragos et al has made a study [44] over the concept of trust. Trust is understood as an action where opposite party is believed acting via the predefined criteria subjective to the entities themselves. When measuring the trust of the entities an abstract concept of trustworthiness has to be defined. Trustworthiness is based on the fact of how the entity has behaved in the past and at the current state. Yet the availability figures, reliability and security evidences has to be evaluated. Evaluation is usually performed for the trust of the devices themselves, for the communication medium and for the trust of the security issues. The trust of the security means the vulnerability of the device against the certain attacks. [44] Trust management is relaying on five criteria when evaluating trustworthiness. Observation is the most important step. In the observation systems, parameters from the entities are monitored. Scoring is done after the adequate information from the entities have been gathered. Scoring can be done for the particular entity or the bundle of the entities. Third step is the selection of the entity based on the scoring results. After the selection is conducted the transaction can commence and more information can be gathered over the functionality of the entity. Finally, at the fifth step, rewarding and punishing of the entities are performed. [44]

In the near future global world will take a leap in device interactions. Now terminology for Internet of Things is evolving rapidly and as an example Cisco has conceptualized a matter of Internet of Everything, in which humans, processes, devices and artifacts are all connected creating added value [5]. As for the comparison Vermesan et al [45] in-

troduces a concept of Internet of Robotics where data combining changes the manner of how artificial entities acts with the humans [45]. Prospective digitalization offers a great opportunity for small and medium sized enterprises for the reason of their agile capabilities for making changes. Although large enterprises are the driving force in ICT innovations, they can convert their methods much slower. [39] Mentioned agile movement of the small and medium sized enterprises has also a down side. Knowledge for using these new opportunities are often unreachable and hard to gain without strong ICT knowledge. There are also islands of knowledge where pure ICT companies have the capabilities of realizing cloud based systems. However, offering the right concepts are cumbersome for them are inadequate to understand the actual requirements of other business sectors. [5; 39] These matters are under a heavy development and as for process data gathering and visualization the practical part proposes a one solution.

2.3 Data transfer methods

The essential concept of thesis implementation part is based on data transfer method for moving the process data from the environment to the cloud ecosystem. Another crucial factor is real-time monitoring of the environment. These two operations are carried out by two distinctive methods. Although variety of data transfer methods exists for cloud based computing and IoT solutions, an approach used here is dictated by the device providers. Section in question takes a closer look for data transmission matters, supported by the source of the process data (ABB Robot).

2.3.1 Representational State Transfer

After internet became reality, HTTP was formed as a standard for transferring data over the web [46]. Selection was mainly carried out for the reason that HTTP messages were able to penetrate company's firewalls. However, HTTP lacked the essential possibility for the use of calling remote objects. As the result, major companies in the field of ICT evolved with a protocol of SOAP (Simple Object Access Protocol). [46] For multiple years, SOAP was kept as the standard for transferring messages [47]. In 2000 Roy Fielding represented in his doctoral thesis [48] a concept of Representational State Transfer (REST) where he conceptualized the web to be constructed of hypermedia systems which are loose coupled with each other. Leonard Richardson and Sam Ruby later refined Fielding's works at [49] to be an approach when designing HTTP-based services. Richardson et al additionally conceptualized the Resource-Oriented Architecture (ROA) which describes the rules when creating RESTful services [49].

In RESTful architecture there are five core constraints and one combined constraint:

1. *Identification of Resources:* RESTful services operate through URI's (Universal Resource Identifier). Thus, all the resource URI's, related to certain application, should be unique, stable and global. [50]
2. *Uniform Interface:* Interactions with RESTful services should be treated with uniform interface structure, which provides the methods for resource serving. [50]
3. *Self-Descriptive Messages:* The representations of resources should give sufficient information for the usage of the current resource. No additional information should be needed. [50]
4. *Hypermedia Driven Application:* Exchanged representations should be linked for additional usage of the original resource. Any entity using the original resource is capable to understand the links through the semantics and is so worth able to continue interactions with provided links [50].
5. *Statelessness:* Interactions between client and server are self-contained. Client state is not maintained on the server; there are no traditional sessions between interactions. State of the client and the server can of course change in consequence of interaction, but following interactions continues from this new state. [50]
6. *HATEOAS:* Items five and six can be bundled for a new concept of Hypermedia As The Engine Of Application State (HATEOAS). Neither client nor server does not maintain the exchange state between sessions. All the necessary information travels inside the individual HTTP message. URI describes the resource and HTTP header and the body contains additional information and the message. Meanwhile, Hypermedia enables the discovering and connecting the services and applications. [51]

Fensel et al delineates RESTful services at [23] to be a combination of multiple Web resources operated between appliances (not by humans) through basic HTTP methods of POST, GET, PUT and DELETE. These methods are the verbs that should be performed for the resource and they describe the actions that are executed; creation, reading, updating and deletion known as GRUD from the initials. [23] When applying HTTP methods there is an important terminology of Idempotence and Safe. Idempotence is a term portraying the methods, which causes the same outcome regardless of consequent requests. Safe, on the other hand, means the methods where the state of the resource does not change between consequent requests. Notions behind these terms are illustrated in the Table 1. The asset of using RESTful services lies both in these HTTP methods [23] and in the ability to use the HTTP status codes [52]. With the status codes client has the tools for interpreting the message by just reading the first three bytes [53].

Table 1. *HTTP methods and basic status codes [54].*

HTTP method	GRUD	Idempotent	Safe	Status code (Collection)	Status code (Specific item)
POST	Create	-	-	201 Created	404 Not Found
GET	Read	x	x	200 OK	200 OK / 404 Not Found
PUT	Update	x	-	404 Not Found	200 OK / 404 Not Found
DELETE	Delete	x	-	404 Not Found	200 OK / 404 Not Found

HTTP protocol is based on the request/response model. Client performs HTTP request for the server addressed with URI. Possible queries for the URI's are separated with question mark. Server processes the request and replies with response message constructed of header and body. Actual data is transposed in the body section of the message, which makes the body significant for the whole transaction. [53] Typically, data is interchanged either with two mainstream formats JSON or XML [23]. XML is a specification for creating custom-based markup languages and it was developed for transferring structured data between information systems [23]. The power in XML lies the extensiveness of the language and in the freedom for the developer to assign own tags for specific purpose. XML being a markup language thus creates significant overhead and requires computing power for parsing and interpreting the message through API's such DOM (Document Object Model). None of which exist while using JSON format. [23] JSON is a lightweight data interchange format operating natively with JavaScript programming language although being a language independent. JSON format requires far less computational power for parsing and interpreting when compared with XML. JSON is also more human readable compared to XML [23]. JavaScript is applied in the practical part. The use of JavaScript lead to obvious and straightforward selection of JSON as for the data transfer format in the REST interface with the ABB robot controller. However robot controller support XML as well [41].

Considering RESTful services used between appliances rather than humans some concepts of descriptive methods for the services are required. Traditional web services implemented with SOAP has been composed with Business Process Execution Language (BPEL) [55]. Another language for describing the web service is WSDL (Web Service Description Language). WSDL is a XML based language describing four elements of web service. First operations, which are supported and operative. Second, the bindings for transport protocol. Third, the exchange format for the messages and finally fourth, the location of the web service. Even though WSDL sketches all the necessary elements for using the web service, WSDL is not meant for ROA based systems and thus all the actions are described in operation-based aspect. Nor WSDL supports for providing Hypermedia links. At the current state WADL (Web Application Description Language) is

the most prominent method for creating the representations for RESTful services. WADL was specifically designed for RESTful services and has the capability of being machine readable in XML form. This method models the resources, which are provided by the service, and additionally holds the information between resources in the form of links. [56] WADL has not gained the mainstream popularity and so worth is only adopted in minority of the services [23].

Verborgh et al states in their Survey of Semantic Description of REST APIs [56] that for the reason of semantics in RESTful services being merely implicit, the developers need to gain additional information over the multiple API's prior to composing a new service. Verborgh et al continues that for the same reason RESTful API's are commonly described with pure textual form [56]. As the case lies also in practical part where ABB Robot controller's REST based interface is portrayed in basic web page only with human capable interpretation [41].

2.3.2 File Transfer Protocol

File transfer Protocol or more commonly known as FTP is an open protocol used for transferring files over the internet. Files, which can be either data files or parameter files for devices. FTP structure is founded on two distinct entities, FTP client and FTP server. FTP servers hosts the files and FTP client connects to the server for either uploading or downloading the files. FTP server can be accessed with specific FTP client software, with internet browser or with command line interface (CLI). While accessing the server with basic internet browser the connection is usually an anonymous. Anonymous however means that server should only provide access to the certain files, which are not confidential. [57]

When FTP was developed, it was not designed to encrypt the data and so worth lacking the according capability. After the initial launch of FTP there has been variations over the original protocol. FTPS (File Transfer Protocol with SSL security) and SFTP (SSH File Transfer Protocol). [57] Both of these connections are secured and so worth used in novel solutions. SSL security stands for Secure Sockets Layer and it is a protocol describing the use of algorithms and variables for encryption of data transmitted and link used for transmitting. The use of according method is based on SSL certificates consisting of public and private keys working together for creating the connection. SSH, standing for, Secure Shell is similar methodology for SSL. SSH provides both encryption of data and authentications of the user at the same time. SSH furthermore has a meaning to describe the solutions and utilities for using the protocol and methodology. [58; 59] These additional variations of FTP prevent the eavesdropping of any unauthorized listeners in the medium used.

FTP connections are based on appointed ports where server and client tracks the incoming messages. Without modifications, FTP server uses port 21 to listen any incoming messages. When requested, FTP server applies port 20 for sending the data to FTP client. FTP client in this case can select the ports to use. Port information is passed with the initial request for the connection. [60]

Because of the popular use of file transfer via FTP, the Unix based operating systems holds a specific version of FTP server called vsFTPd. vsFTPd runs as daemon (Unix OS program running privately on the background without user intervention) in Unix based operating systems such as Linux. The power behind the vsFTPd lies in its security and stability augmented with its capabilities in performance. [59] Multiple are found on-line for the configuration of this particular FTP server and due to these stated matters vsFTPd was selected to be the utilization of FTP server in implementation work of the thesis.

2.4 Cloud based ecosystems

In this chapter introduction and comparison for significant operators in the field of cloud computing and IoT Dashboard platforms are represented. The following services and software's has been preselected through them known existence and capabilities. According to report from The Research Institute of Finnish Economy merely on the field of IIoT 50 different platforms existed in the year 2014 [5]. At In the future some of these platforms will polish to be the ones for future use. Others will diminish or change their vision [5]. For the implementation, there was a requirement for platforms and services that, in high probability, still exist in the future. To get the more precise overview for the current state of the cloud ecosystems one section of this chapter is dedicated for presenting the alternatives for major players.

2.4.1 Amazon Web Services

Amazon Web Services (AWS) started in the 2006 with a provision of IT infrastructure for various fields of business. At the time, these utilities where called web services. Now according services has adopted a new name; cloud computing. AWS had an idea of providing infrastructure by which the customers can replace their own up-front expenses with low costs. In their vision, costs are build up from the time of using the services. AWS started to use the term pay-as-you-go. AWS realized the potential behind the ideology where customer's own infrastructure does not lay on the way of future growth. Customers can automatically expand their own services through automated sequences inside the AWS cloud services. [61] AWS has developed a global, scalable, reliable and low cost infrastructure operating at 14 geographic regions in 38 availability zones (datacenters) with additional expansion in coming years. [62] AWS provides over 70 different services from which the customers can selected the ones for their solution

[61]. According to Gartner Magic Quadrant (Figure 5), being 10 years in the market AWS has gained a notable lead compared with rival service providers and up comers. [63]

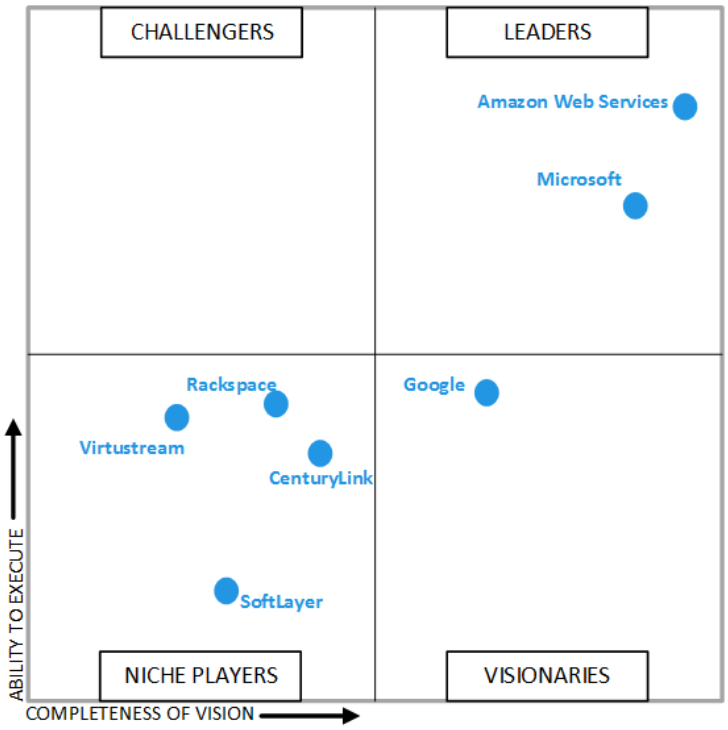


Figure 5. *Gartner Magic Quadrant representing cloud IaaS major players, adapted from [63].*

For the new users Amazon Web Services offers a concept called Free Tier, starting from the sign-up date. When customer is registered through AWS portal user is granted one-year free usage of certain services with certain limitations. Free Tier covers roughly half of the entire product line and all the major services are included. Free Tier is divided into a monthly usage and any use over the Tier is charged separately. In most incipient trials and concept designs for the usage of the cloud services Free Tier is highly adequate. After Free Tier year customer is charged according to AWS standard pricing. [64] For the marketing purpose, AWS has developed quite hooking advertising system.

AWS provides extensively layers of security for their customers through years of security development. Customer cloud is protected with an AWS concept of Virtual Private Cloud (VPC) incorporated with firewalls. Customers are eligible for building their own subnetworks inside the VPC environment and ruling their gateways for internet accessing. All the actions inside the cloud infrastructure are encrypted with TLS (Transport Layer Security). Customers are in addition able to construct security groups providing the IP (Internet Protocol) address rules for inbound and outbound traffic. Accessing with API (not via AWS portal) to the different services, connection is verified with identification number consisting of numbers and letters of capital and lower case. Ac-

cording concept is called AWS IAM (Identity and Access Management). Much of the security issues and details are held in secret. Amazon does not provide the entire description of their security mechanisms. [65] Amazon services can be kept reliable through their long period on the market and positive feedback found quite easily from couple of searches from the web. Amazon states having redundant power sources and independent networking and connectivity solutions housed in separate facilities within different availability zones [62]. In addition, equally important matter is the concept of how and where the customer data is stored. Amazon Customer Agreement [66] assures that the customer data is always stored and kept in the availability zone (datacenter) which customer has made the selection. Data is newer moved, without customer clearance. Thus, in the case when data is stored in EU economical region, data will always stay there. Although AWS has the disclaimer, stating that in illegal use cases the data is handled according the legal regulations, although this action is informed for the customer in advance. [66]

At the time of online tutorials and forums, AWS has found the way for mercerizing their new features and provide assistance for building solution(s) with their platform. They offer a great deal of their own webinars via Amazon Web Service web pages, as well almost regulatory at the modern ages, via YouTube channel. Individual users and customers are also discussing the features and possibilities of AWS platform at online forums. These discussions provide a fundamental ground for any new customers to get started.

For accomplishing the tasks set forth for this thesis multiple AWS services should be implemented. Figure 6 portrays the overview of the required services. All these services are part of AWS Free Tier offer [64] and multiple tutorials over these services can be found for getting started. As the robot supports only three possible data exchange methods (see Chapter 1.3 for details), some decisions need to be made. Depending from the data gathering frequency, timestamping might be fuddled and internet connection might get overwhelmed if each triggered data entry is transferred independently. More convenient way is to first gather the data inside robot controller and transpose it after the process is finished. Real-time monitoring on the other hand should be conducted with specific intervals from controller. FTP is a convenient way to transfer the data after the process is finished. Amazon Elastic Compute Cloud (EC2) [67] with an instance of Amazon Linux operating system can hold a FTP server which robot controller can access with its client [68-70]. EC2 can additionally hold Node.js server [71] providing a platform for executing REST interface. By way of REST service real-time monitoring can be achieved.

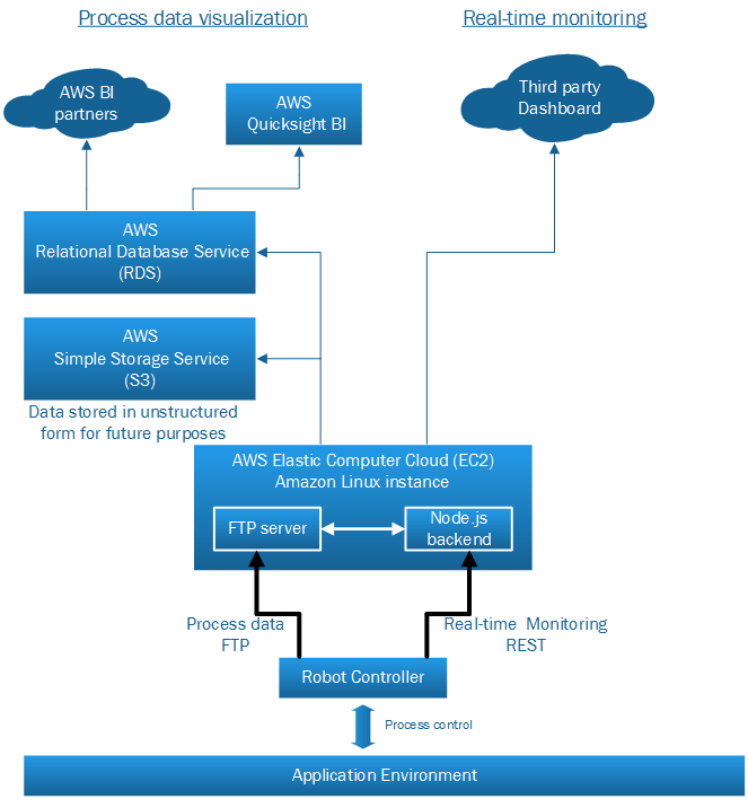


Figure 6. *Amazon Web Service infrastructure for implementing the data gathering and visualization*

Amazon Web Service does not include any Dashboard solution for illustration purposes of the real-time customer data. AWS has an IoT service yet it is designed for making the D2D connection rather than visualizations [72]. From these grounds, variable data needs to be transferred to additional frontend solution. Process data is received in text file format, which can be moved into AWS Simple Storage Service (S3). S3 acts as permanent deposit of the original text file. Meanwhile text file can be parsed and data can be moved into AWS Relational Database Service (RDS) [73] holding multiple different database options. Amazon Aurora, PostgreSQL (Structured Query Language), MySQL, MariaDB, Oracle and Microsoft SQL Server [73].

For making the analysis and visualization of the gathered process history data, customers can rely on AWS QuickSight service [74]. QuickSight is an AWS cloud powered business intelligence (BI) service through which customers are able to make business insights over their data and ad-hoc analysis or visualizations. QuickSight is capable of searching the data straight from all AWS cloud storages, including S3 and RDS. The engine behind the service replays all the data and forms a best-fit conception which kind of analysis and visualizations user would like to use. Service further more offers this solution for the user, although user can afterwards make own adaptation of the data visualization. QuickSight has one weakness for not having an automatic report creation possibility. [74] Almost all the areas of business are depending in some sort of automatic or semi-automatic report creations prospects. While being a decade on the market

AWS has gained a substantial amount of partners integrated with AWS services and so worth providing third-party business intelligence solutions [75].

2.4.2 Microsoft Azure

Microsoft Azure, originally named Windows Azure [76], was launched in 2008. Microsoft desired to join in the rivalry of the cloud based services. Amazon had started their own counterpart few years earlier and Microsoft realized the market opportunity's as well. In 2014 Windows Azure was renamed Microsoft Azure [76] and it had already gained substantial position at the cloud market. However, the starting point was something different. Azure was originally built with the Silverlight ecosystem [77]. In 2011 there was a change in the management of the Azure division and platform was changed for much lighter HTML5 (Hypertext Markup Language). From this point Azure started the eminent growth. [77] Azure utilizes exactly the same business model with AWS, pay-as-you-go. Azure operates on 30 regions (datacenters) at the moment and they have 8 more on the horizon. [78] Azure provides 100 different cloud based products for their customers [79].

Microsoft has divergent pricing method for new users when compared with AWS. Azure offers 170 euros of credit which user can utilize in any of their cloud service products [80]. Method can be considered with the use of Azure pricing calculator [81]. In case of implementing the thesis practical part with Azure services, amount of credits provided for new customer would withstand one and half months of executing the solution [81]. This is a considerable difference when compared with AWS Free Tier offer. However, Azure van compete with the mentioned fact that credits are valid for any service in their product line.

Microsoft has a well-known experience over the security and reliability issues working with their global product line. Microsoft as an operator itself is kept in value and with high probability; the company will remain existence for years to come. Azure is a direct competitor for AWS and has already gained considerable amount of partners [82] and customers [83]. Meaning that closing down Azure platform would extensively affect their brand and image as a company.

Microsoft has a concept called Security Development Lifecycle that is a software development process where security issues and compliances are addressed during the development project. *Assume Breach* is a concept through which Microsoft performs testing for the services. Team of trained security experts simulates real-time attacks against the network. [84] At the level of physical security, Azure datacenters are prepared for sudden power loss and any physical intrusion or network outages. Customers are eligible to build their own isolated networks within their regime and operate between these regimes via private IP addresses. Virtual Private Network (VPN) connections between

Azure cloud and customer location are also conceivable. Azure has committed for EU Model Clause. An EU data protection law that regulates the data provision outside the EU economical region. In addition, Microsoft Azure is acting along US-EU Safe Harbor Framework and US-Swiss Safe Harbor Program. Microsoft was one of the original subscribers of ISO/IEC 27018 international guideline for practicing cloud privacy. [84] From the Azure Privacy Compliance document [84] a clause is found for stating that the data location remains on the region where customer has originally made the deposition. Microsoft move the data inside the region for another datacenter for redundancy purposes. However, data never leaves the economical are where it was initially stored.

Microsoft is advertising their Azure concept with extensive webinars, online tutorials and brochure marketing material. Azure team has made extensive work with their portal solution. Customers can easily select the service they need and deploy it into use. Portal interface is based on method where new pane opens on the right hand side of the source pane. Even though making the user experience friendly, not all settings are found in the same amount of time compared with AWS. Azure manages with the appearances, yet losses some on the demanding settings. Individual studies and customer tutorials are not at level compared with AWS. Years on the market and the development from the year 2011 has helped to catch the pit. However, there is still road ahead.

In the circumstances where practical part of the thesis should be implemented with Azure platform, multiple services has to be commissioned. Figure 7 illustrates the concept where thesis implementation objectives are conceived with Azure services. Initial limitations and assumptions can be revised from Chapters 1.3 and 2.4.1. When the process in the application environment is finished, the robot sends the process data via FTP. Azure Virtual Machine service can host Ubuntu Linux [85] with FTP server. One tutorial gives instructions for proFTPd server [86], yet vsFTPd should work equivalently. Reason for, vsFTPd is native FTP server for Unix based operating systems. Node.js runtime environment can be installed on Azure Linux VM. Thus, JavaScript based execution for arrived process data file can be performed. [85] Customer program execution is able to move the text file into Azure Blob Storage for permanent storing.

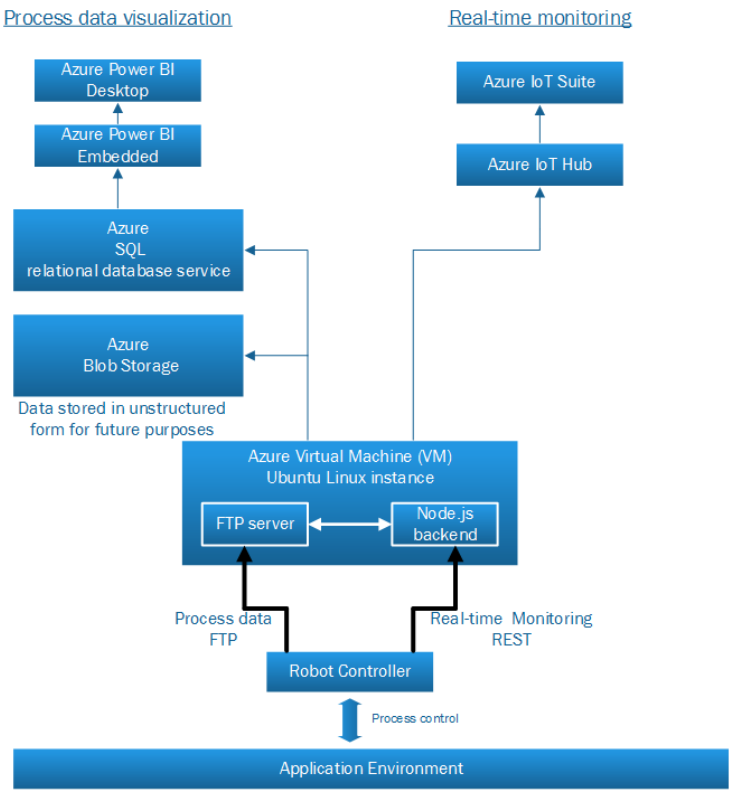


Figure 7. *Microsoft Azure infrastructure for implementing the data gathering and visualization*

Real-time monitoring can be achieved by setting up a REST service inside Azure VM. According service acts as an interface for requesting the desired variables from robot with certain intervals. Concerned REST service is again able to communicate with Azure IoT Hub. IoT hub is a platform for connection of IIoT and IoT devices into Microsoft ecosystem. Devices can connect with HTTP or Message Queue Telemetry Transport (MQTT) interface or via gateway in the case where no internet connection is available within the device. [87] Three programming languages are presented in the Microsoft tutorial for creating connection with IoT Hub, C#, Java and Node.js [88]. Custom App Service can act as a gateway between robot and IoT Hub providing the end-to-end connection. On top of IoT Hub, Azure provides IoT Suite through which users can visualize their data in real-time. Historical data visualization is additionally available, from the time when data submitting started. [89] Both IoT Hub and IoT Suite are applicable of communication bi-directional with the device [87].

Azure Blob Storage [90] is a service where customer can store any unstructured data, such as text files. Unstructured data could be moved into Azure SQL relational database service. From this point, data can be visualized and analyzed in detail. Final step in the implementation would be commissioning the Azure Power BI Embedded and Microsoft Power BI Desktop. Power BI Desktop [91] is one of the newest Microsoft products for creating Dashboards analyzing and visualizing business data. Power BI Embedded acts here as a gateway for retrieving the data from various sources, such as Azure SQL Da-

tabase [92]. Through Power BI, customers are applicable for making embedded visualization within their web and mobile applications [92].

2.4.3 Google Cloud Platform

Google is a corporation with well-known name. Their dominance in mobile operating systems, civilian email & data storing solutions and video sharing services is almost unbreakable. In year 2008 they released Google App Engine [93]. Service was similar with AWS Elastic Beanstalk or Azure App Engine. An ecosystem possessing runtime environment for various programming language support, making the customer application deployment easier. [93] Google had noticed the potential in cloud computing and involved themselves to the race of gaining the market shares. Now Google Cloud Platform (GCP) operates on 5 geographic regions and has total 15 zones available within these regions [94]. Zones are physical datacenters inside the geographic regions and different zones are connected with fiber optical communication for making the redundancy behavior easier. GCP has additionally 8 regions on the verge of the launch state. [94] Google offers 52 different services from which the customers can select the ones for their purpose [95]. Google has adopted similar pricing than their rivals. Per-minute paying is the term they use [96]. At the Gartner Magic Quadrant [63] GCP is set in the class of visionaries. When studying Google marketing material and tutorials over the different products they offer through their Cloud Platform, this categorizing comes clear. One example can be noted from their launch of Node.js platform in Google App Engine [97]. The level of fine-tuning and readiness in the event does not reach the level of their competitor's similar events.

For the new user of Google Cloud Platform they offer 300 dollars' fee, which can be used for any of their Cloud Products. Fee is offered for 6 months or when the amount is consumed. Some of the products are outside the offer. These products are offered as without no cost for new trials. [98] GCP pricing calculator gives the details over the convenience of the offered credits. Building similar infrastructure as with AWS or Azure, GCP would utilize the fee slightly over three months after which the usage of the GCP would cost around 90 dollars per month [99]. Google offer performs slightly superior compared with Azure yet loses the rivalry with the AWS in here.

Google has published a Whitepaper [100] over their security and privacy issues. According to this whitepaper, they perform extensive background checking for their personnel working with the security and privacy issues. Whitepaper states that collaboration with the security research community has being one of Googles elementary operations during the years at market. For making the security breach hunt more inviting, they have started Vulnerability Reward Program. Current programs offer reward for any individual person finding a possible breach. Google exploits TLS encryption in the data transferring and has initialized the use of 2048 bit RSA certificate to make the data

transmission inside cloud ecosystem safer. Physical datacenters are highly guarded and implemented with multiple layers of security. Final entrance inside the data center is protected via security badges and biometrics. Uninterruptible power sources acts as backbone for the servers. [100] For the privacy and data location Google announce in their Terms of Service [101] that customer data is stored according to their Service Specific Terms [102]. Additional search through these Terms reveals that customers can only selected the geographical region where the data is stored. Google makes the final decision which datacenter is employed at the time. [101; 102] Nevertheless, customer can make the choice for specific datacenter via Google portal. This mismatch feature can puzzle the customers for they can make the choice that does not take any notice.

Google has operated at the Cloud computing market for almost ten years now and is competing with AWS and Azure [63]. However, Google has not found the way to perform equally with other major players. Google does not reach the level of AWS nor at the level of Azure with their tutorials or getting started guides. Minor amount of implementations can be noted from the amount of customer's posts and inquiries on the online conversational forums and tutorials. Some of the Google products are working on the beta version, expecting additional feedback from the users. Practice, which might banish the corporation level customers. Nevertheless, Google can compete with their massive computing power and their study on the artificial intelligence or neural network research [103]. Expertise, which has induced customers from the fields where vast number of computational power is required. Visual special effects companies, for an instance. [103]

For reaching the outcome thesis practical implementation, multiple GCP products should be put into operation. Limitations and assumptions can be revised from Chapter 1.3 and Chapter 2.4.1. Most of these products are part of the Google 300-dollar fee and one products is totally without any charge for being at the beta state during the moment. Figure 8 illustrates the concept design with Google Cloud Platform products. On the fundamental level, GCP Compute Engine would be commissioned. Compute Engine can hold a virtual machine instance launched from multiple operating system [104]. From these systems, Ubuntu Linux would be the most suitable. Reason for this is the possibility to launch Unix based FTP server, vsFTPd. Ubuntu Linux is additionally capable of hosting runtime environment for Node.js [105], among others. When the process on application environment is ready, the robot would send the process data file to FTP server. Updated folder can be noticed and the file could be transposed into GCP Cloud Storage [106]. An object based data storing service. At parallel action, data can be stored into GCP Cloud SQL service hosting MySQL relational database [107].

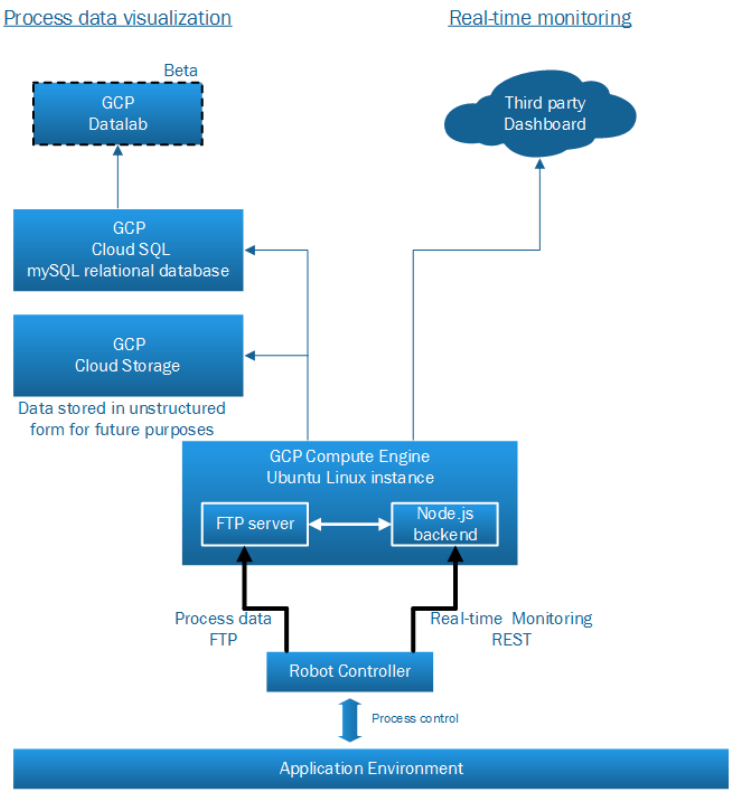


Figure 8. *Google Cloud Platform infrastructure for implementing the data gathering and visualization*

Here lies the crossroads between competing service providers (AWS and Azure). Data is now stored inside Cloud Storage as raw material for permanent keeping and for future use. Equivalent data is structured inside SQL relational database. Sequential step should be visualization both process data and real-time data. Fetching of real-time monitoring variables is able to treat with a REST interface running on Compute Engine. However, Google Cloud Platform merely holds a Cloud Datalab visualization tool, still at beta stage [108]. Furthermore Cloud Datalab is actually a platform build on Jupyter [108] a web application for sharing diaries that holds equations, analysis or visualizations [109]. Cloud Datalab support multiple programming languages for making customers own visualizations i.e. programming of the visualizations need to performed conventional web application manner [108].

Visualization cloud be done with Cloud Datalab or the data could be provided for some third party solution, making the visualizations and possible analysis. Real-time monitoring could be alternatively handled with some third party Dashboard solution supporting convenient data transposing interface, REST for an instance.

2.4.4 Alternatives

This section covers the alternatives for major players in cloud providing cluster. Minor operators are shuffling the field and wresting customers from their giant opponents. Later on at this section comes the review over open source methods for incorporating cloud computing. At the end of the section, there is a comparison against the private cloud where all the functionalities are hosted inside the company network.

Comparing with AWS, Azure and GCP, SoftLayer has the potential for contending with these operators in near future. SoftLayer was acquired by IBM in 2013 and is now a subsidiary of IBM. SoftLayer provides virtual servers and basic hardware servers for their customer's use. They, in addition, have solutions for data storage and networking services. [110] Rackspace (creator of OpenStack) and CenturyLink have slightly different aspect for cloud service provision. These companies are acting as integrators and administrators providing managing services for businesses using major cloud platforms. [111; 112] Company called Virtustream has focused their business model for providing cloud platforms dedicated to Enterprise Resource Planning (ERP) sector. Company offers cloud platforms for SAP, ERP, Oracle, Git and Apache among many others. [113]

One rival on the cloud service provision field is a Finland based company UpCloud. UpCloud has developed concept called MaxIOPS [114] a disc management technology making input and output operations faster when compared to conventional cloud providers. UpCloud infrastructure is scalable according to customers' needs and customers are capable of selecting the platform from their own instance or one from UpCloud archives. [114] Concerned company can play a major role in the future and compete by way of trust and promise of data integrity and privacy.

Cloud consumers are not constrained to use a commercial version of cloud computing platform for there is an alternative; open source cloud platforms. EUCALYPTUS, OpenStack and OpenNebula are the most progressive ones in the field. EUCALYPTUS is derived from *Elastic Utility Computing Architecture for Linking Your Program To Useful System*, which is an open source solution developed in University of California-Santa Barbara. Advantage in the EUCALYPTUS project lies in the technology where AWS compatible API accesses both AWS EC2 compatible counterpart, EUCALYPTUS Computing Platform and AWS S3 compatible EUCALYPTUS Cloud Storage. [115]

Even though EUCALYPTUS has a founding region of contributors, the most rapidly evolving open source cloud project is OpenStack. OpenStack originated from collaboration between National Aeronautics and Space Administration (NASA) and Rackspace. NASA provided their Nebula solution, renamed as Nova, for computing resource and Rackspace involved with their Object Storage later renamed as Swift. OpenStack possess an Image Service called Glance. Glance is used for retrieval and looking up images

from VM's. Such as the case was with EUCALYPTUS, OpenStack has the feature where accessing the cloud can be managed via AWS compatible API's. Thus, solution build on AWS ecosystem can be deployed to OpenStack. [115] A vast amount of developers can cover multiple fields of technology which makes Open source methods a remarkable assets for developing cloud computing technologies suitable for particular field of industry.

Third alternative in the list of open source cloud computing project is aged solution first introduced by Liorente, I and Ruben, S back in 2005 in their research project. When the solution was released in 2008 it was titled as OpenNebula. Cited project has a different approach than two mentioned above. OpenNebula is used for virtualization of physical resources located in data centers. These data centers are generally private ones although OpenNebula holds an option of exposing its interface when functionalities with public clouds are conceivable. By way of interface exposing OpenNebula holds a support for implementing hybrid clouds. [115]

Comparing these mentioned open source solutions, the task comes a bit cumbersome, when OpenNebula works on the different abstraction level compared with others two. The most convenient way for comparison is to use some notable commercial cloud provider as reference. Amazon Web Services is selected for the reason that background of questioned provider leans back to the time when referred projects has started. Both EUCALYPTUS and OpenStack holds similar features compared with AWS EC2 and AWS S3. These open source projects both include AWS compatible API's. Within OpenNebula there is also an AWS compatible API for building hybrid clouds although OpenNebula works on different grounds. In addition OpenStack developed their own native RESTful service API for accessing the resources [115]. This RESTful API was later commissioned in Duan et al study where a hybrid cloud was introduced with OpenStack Swift as the object file service [7].

In the case where pure private cloud needs to be considered a survey over the companies own IT infrastructure has to be conducted. Of course, the solutions in here varies extensively between the companies. However, next there is an example of the one possible solution. Some companies might offer platform services internally with IaaS, PaaS or SaaS methods. Smaller companies might only have tiny ICT team without any ready-made packaging of services. Either way after the platform is initialized the actual solution needs to be divided for backend and frontend levels. Backend provides the data storing and parsing while frontend handles the real-time monitoring and visualization of gathered data. Described method requires more expertise in multiply fields of technology when compared with commercial cloud services. One possible structure of the implementation is portrayed in the Figure 9.

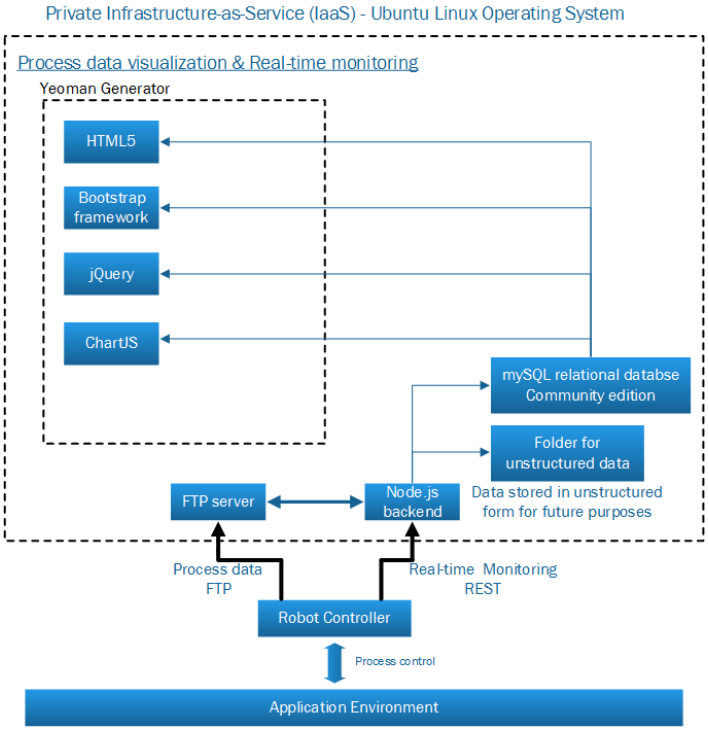


Figure 9. *Data gathering implemented with private cloud infrastructure*

2.5 Dashboard solutions

Rise of the IoT and IIoT paradigms has brought on the appearance of the Dashboard solution for making the visualization, analyzing and reporting much more straightforward when compared for traditional web page based applications. According to Research Institute of Finnish Economy cloud based systems and IIoT demands high expertise at the field of ICT [5]. Yet, there is a change happening. Cloud service and IoT platform providers are improving their framework to reach the level where implementations require much less of high-end expertise. Dashboard solutions are the final step for providing the tools for more cost-effective implementations. In this chapter, potential alternatives are studied.

2.5.1 Wapice IoT-Ticket platform

Wapice Ltd. is a Finnish based company, which operates as technology partner with industrial business sector. They provide custom software and embedded electronics solutions for improving their customer's competitiveness. In addition, for their custom solution they have their own products. IoT-Ticket platform is one of them [116].

IoT-Ticket is an IoT platform, which provides data gathering, supervisory monitoring and controlling through internet based Dashboard User Interface (UI) solution with analytics and reporting capabilities of gathered data [117]. Data gathering can be handled either via Wapice proprietary WRM (Wapice Remote Monitoring) device, by REST

interface on top of OPC (OLE for Process Control) Gateway, customers own REST client, Libelium interface or Wapice ready-made developer libraries. Building of the controlling and monitoring Dashboard is conducted by "drag 'n drop" principle via *Interface Designer* tool running on web page based portal. This means that no software needs to be installed on customer's workstation. Elements on the Dashboard and the gathered data are then interconnected with *Data Tags (*referred also as datanodes*)*. Second tool is web based portal application called *Dataflow Editor*. *Dataflow Editor* is a Wapice in-house design tool operating on graphical block programming basis, based on the IEC 61131-3 standard. *Dataflow Editor* is used for programming interactions with data tags. [117]

Wapice *Report Editor* is another solution integrating with IoT-Ticket Dashboard. Report Editor operates on web page based portal solution with "drag 'n drop" method. Creation of the reports takes place from the gathered data inserted with static material available in each of the reports. Static material can be company logos and manufacturing line indicators. After the layout of the report is designed, reports with similar template can be automatically triggered or manually requested. Another web page based solution integrating with IoT Dashboard is Wapice *Analytics Tool*. With the questioned tool, gathered data can be analyzed for better understanding both the process and the phenomenon affecting to the results. *Analytics Tool* includes correlation analysis with additional analysis managed through line, scatter, map and bar plots. Analytics tool offers the users to perform an abnormal value detection by using the curve fitting algorithms. For the implementations not getting enough information out from *Analytics Tool*, Wapice offers an interface for the R. R is a statistical computing language with extensively analyzing capabilities and options. [117]

IoT-Ticket has the advantages based on the entire line of solutions. Using same application users are able to access monitoring and controlling of the appliances with the addition of using reporting and analyzing tools. Another asset lies in the method of data gathering. Either user are can purchase ready-made WRM device, use their own electronics with REST interface, or order proprietary electronics from Wapice including IoT-Ticket interface. The whole application package is user friendly. With basic "drag 'n drop" architecture the learning curve for making new solution or modifications to existing ones is kept at the low level. Customers can modify the appearance of the Dashboards to match their company's brand image. [117] Even though IoT-Ticket possess multitude of benefits the weak point of the solution is the lack of the custom code execution platform. In some cases, data processing before insertion to IoT-Ticket could have additional value. Additionally, execution of custom code triggered by Dashboard solution could be convenient. At the current state, customers are forced to execute the code on another cloud platform and handle the data transmission with suitable interface. However, Wapice is concentrating on different section of the IoT market. Leaving the code execution to the hand of operators from different sector.

2.5.2 Freeboard.io

Freeboard.io is an open-source IoT Dashboard platform with quite modest pricing structure. Without any charge user can make unlimited number of public dashboards, open to entire world. For making private Dashboards Freeboard.io charges 12-dollar fee per month. For this payment user is yielded with five dashboards located behind the password. Dashboard development is based on the widgets, which makes it easy and swift. Freeboard.io uses integration with dweet.io for connecting the devices, yet it also has a REST interface capability. [118]

REST interface and "drag 'n drop" simplicity makes the Freeboard.io one of the most prominent alternatives used in frontend technology. However, Freeboard lacks the possibilities for report creation either on automatic mode or manually. This is due to the fact that Freeboard.io acts solely on the real-time basis. [118] Report creation should have database connection or capabilities. Freeboard.io is an open-source solution, which effects for the supporting means, tutorials and blogs on rather modest state.

2.5.3 Ignition IIoT

Ignition is a server based SCADA system with HMI (Human Machine Interface) capabilities developed by the company called Inductive Automation [119]. Ignition works on the centralized manner in which all the devices and applications inside the factory can connect and deliver the data for visualization through the HMI. Data is stored in SQL database by Ignition adapters and HMI development is conducted with Web page interface mainly by the "drag 'n drop" technique. [119] Although Python scripts are supported for operations that are more exclusive. Software runs on Java platform meaning that HMI can be deployed to each device supporting Java Runtime System [119]. Mobile devices are supported as well. Mentioned database connection enables reporting and history diagram inspection. Real-time monitoring features are incorporated with HMI system. Feature, which makes the Ignition system quite intriguing for the factories and manufacturing companies, is the pricing model. In their pricing one license fee makes possible of deploying endless number of client HMI's, even with off premises distant locations. [119]

The appearance of the concept Internet of Things has awaked the interests inside Inductive Automation as well for they have developed Ignition IIoT platform making the data delivery for their Ignition SCADA software. Ignition IIoT utilizes MQTT protocol, which is gaining rapid interest with machine-to-machine connections. MQTT is preferred in many cases for its lightweight overhead, scalability and security facilities when operating with IoT and IIoT. MQTT has also better usage of bandwidth due to its Publish/Subscribe protocol ideology compared with polling protocol structures. Ignition IIoT can operate with three different principles. First public cloud based basis, pure pri-

vate basis or hybrid cloud bases in which MQTT servers are running on public clouds. [120]

Advantages of the Ignition IIoT lies in the background of the software itself. The whole concept is actually on add-on for Ignition SCADA system. For the users of Ignition SCADA, Ignition IIoT could be an obvious choice. Disadvantages are forming from the high initial price of the software [121] and the deprivation of the REST interface [120].

2.5.4 DGLogik IoE platform

DGLogik is a company providing customizing solution for visualizing the Internet of Things device data. DGLogik uses the term Dashboard for their creations of the user interfaces and in addition refer to the IoT as the Internet of Everything (IoE), where users are capable of analyzing, measuring, managing and controlling the devices from any location. [122] In the interest of Dashboard technology DGLogik methodology is based mainly on their DGLux5 application developing environment. DGLux5 is HTML5 compatible as the number 5 indicates, making the applications natively responsive for mobile devices. [123] Finished applications are adequate for performing real-time monitoring of factory functions either on the shop floor or at the production line level [124]. DGLux5 additionally possess analytic capabilities by providing formula creation and manipulating the data before inserting into visualization [125].

DGlux5 operates on “drag ‘n drop” fundamental without familiarizing the developer with the actual code behind the widgets. Visualizations, charts, graphs and data flows are handled with appending these widgets and connecting them with elastic wirings. Finished solution runs solely on any of the client side web browser, which supports HMTL5. Described method releases the server (backend) side for handling the incoming data and providing it for client (frontend) solution. [123] DGLux5 comes with Apache Tomcat 7.0.55 web-server that makes the server applicable for installation either an Windows, OSX or Linux systems. Data inbound can be treated via various different methods, MQTT being one of these. However, DGLux5 in addition offers REST interface for data transmitting. [123] DGlux5 pricing is quite modest when compared with current rivals as Ignition IIoT [126]. This makes the software attractive for new installations of IoT Dashboards.

2.5.5 Conventional Web Application

Web application, modern term for web page, development has evolved considerably after the initial versions of web pages, build with basic HTML language. New customs and new methodologies are taken into practice. Following section takes a review over the modern technologies from the aspect of building IoT Dashboard solution. Still keep-

ing in mind that in IoT concept web apps can be perceived to be conventional when comparing with IoT Dashboard solutions.

Modern web apps are based on three building blocks. First, there is a traditional HTML language. Second, CSS (Cascading Style Sheets) [127] is the tool for describing how the elements in the web page should appear for the user. Same CSS file can be used in multiple web pages. Method that makes it easier for developing similar layouts. CSS was designed to fix the deprivation in HTML. HTML only incorporates the content of the web page, not the layout of how the information should be represented. [127] Third, JavaScript, a scripting language which runs natively in all modern web browsers. In web application development JavaScript has gained reputation through the library called jQuery.

When starting a development of state-of-the-art web app, the designer has various methods of how to proceed. One possibility is to use Yeoman generator ecosystem. Yeoman is JavaScript package that builds developer ready infrastructure for writing the actual application [128]. Yeoman provides automatic web app folder structure and required files inside these folders. When installing the ecosystem through Yeoman user can make a selection for frameworks as Sass, CSS or Bootstrap. Sass is an extension language for writing CSS files more efficiency [129]. Whereas Bootstrap is a framework for creating responsive and initially mobile based applications [130]. Mentioned term, responsive, indicates that web app is especially designed for mobile devices. Elements on the page can scale, hide or move in relation to the end-device screen resolution. From Bootstrap web page [130] developers are able to find hundreds of examples of different elements and code for creating these elements inside in questioned web app.

Above-mentioned jQuery is a library for writing JavaScript web apps more efficiently though convenient event-handling, animation, HTML manipulation and AJAX (Asynchronous JavaScript And XML) programing [131]. AJAX is especially important with RESTful services as AJAX concept can be stated to follow RESTful service design guidelines. AJAX commands are sent with HTTP request and HTTP response alters the web page through JavaScript commands. [132]

Elements of buttons and meters can be taken care with mentioned technologies, although for tracing real-time and off-line variables, concept of charts are needed. Current web page applications can have charts incorporated with multitude of techniques. The most sensibles are Chart.js, D3.js and dygraphs. Chart.js and dygraphs are open-source solutions that can operate on very basic chart tracing. D3.js is a very high-ended data visualization toolset with steep learning curve. However, it is open-source and does provide visualized examples. [133]

It can be stated that the learning curve for building IoT Dashboard with web app is much steeper when compared with ready-made IoT Dashboard applications. Yet, with

sufficient expertise, web app enables more features that can be accomplished with built in platforms. The final selection on technology is a compilation of expertise, use case, scale of the system, available resources and modification availability.

3. METHODOLOGY

Following chapter describes selection for the technologies used in the practical work. After which comes the detailed descriptions of these solutions. On the higher abstraction level, implementation is divided in three independent parts. First, is the application environment layer where the actual process takes place and where all the physical devices are connected through interfaces. This layer is also the source for the process data, which is the main interest of the implementation. Application environment was already stated when starting the work of the thesis so there was no influence from the work to physical devices. Thus, cloud services needed to be applied to these already existing devices. Regardless of sketched situation application layer plays significant role in the implementation and methodologies in the devices are illustrated in the following chapter. Second comes the concept of backend technology. Amazon Web Services plays the role of processing the data and providing it for the visualization, taking place at frontend technology; part three of the implementation.

Segmentation for backend and frontend technologies was conducted for the hypothetical future work where either one of the technologies could be replaced with parallel technology. Another reason was that either one could be altered without affecting to the execution of the other. Thus, Software-as-Service (SaaS) principle was adopted. For the same reasons the programming language was selected to be JavaScript and the backend server runtime as Node.js. Both of these are gaining more popularity and they possess a high probability for having the support on cloud platforms in the future.

Following chapter starts with the technology selections and continues for the description of the application environment. After which the focus turns for the backend technology. Finally, frontend technology is detailed.

3.1 Technology selections

Technologies used in the implementation are selected through the theoretical study from the Chapter 2 of the thesis. Final selection was quite straightforward due to the fact that practise of weight factors was implemented for the most important features of the technologies. These most important features included, pricing of the technology, future existence, reliability of the service provider, data privacy and service security, low learning curve, support for Node.js execution, REST service support in Dashboard solution and extensive software support either through direct contact or through tutorials and online forums. Current section starts from the selection of the programming language

and continues for the selection of the cloud technology and final to the Dashboard technology. Method of tables are adopted for making evaluation.

3.1.1 Programming language selection

All major cloud service platforms (AWS, Azure, GCP) support multiple programming languages for server implemented code execution. In the implementation, REST services for the robot controller and for the frontend solution plays a significant role. From the previous experience, there was a knowledge that JavaScript with Node.js environment has an extensive support for REST services through additional package and so worth both JavaScript and Node.js was kept as a preferred programming language-runtime compilation. Robot controller has a Digest authentication for the REST interface and Node.js *request* package additionally incorporates the concerned method. From these grounds, JavaScript with Node.js server environment was selected for the programming language.

JavaScript was originally developed for a scripting language, rather than pure coding language, used in HTML web pages. JavaScript is executed at the client side of the service. Thus, only a few lines of code is needed to command the client browser to perform the required actions. [134] Another design rule which makes the JavaScript ideal for HTML pages is the asynchronous execution method. Asynchronous means that various code executions are performed simultaneously compared for synchronous method where code is executed line-by-line. Consecutive line waiting for previous one finished. In the design principles of JavaScript, there are no fundamental limitation that it could not be used at server side as well [134] thus Node.js was developed.

Node.js adopted Google Chrome V8 JavaScript engine and implemented a server side execution. Node.js is an event driven and non-blocking method for building scalable server side solutions. [135] Node has gained extensive popularity in the recent years and vast number of independent developers are participating in Node package ecosystem, *npm* (Node Package Manager), for making their own code packages performing divergent actions. Npm is mentioned as the world's largest open source library. [135]

3.1.2 Cloud technology selection

For the cloud service technology and provider there was no preferred selection when starting the evaluation. The whole process was handled through initial requirements (see Chapters 1.3 and 2.4.1 for more details). Open source cloud computing services where left out from the evaluation. There were two reason for such action. First, the future existence of the service had set a great value. Second, the readiness for supporting the commercial (Small and Medium sized Enterprises) solutions was one of the primary goal. Private (on premises) cloud solution was left out because in the work description,

public clouds were set to be the technology to use. UpCloud will be one potential alternative in the future. Now UpCloud provides solid application execution environment with high speed hard drive system. However, no fee for new customers are offered and product line is not yet equal with other rivals. That is to say, evaluation would be cumbersome with no similar products available.

In the evaluation table, each service provider can score from 0 to 100 on each feature. If all the providers manage themselves equally, everyone is provided with same value. Score values are weighted through factors decided according to the importance set forth in these features. Pricing, learning curve and existence of the service in the future where the main objectives. This appears in the evaluation table where 20 % factor is provided each of these features. When something as essential as production and process data is in question data privacy was additionally provided with same 20 % weight factor. In the case if some provider would score 100 on each of the feature, according provider would get final score 100 which is the maximum. After the theoretical study and the assessment of Table 2. Amazon Web Services was selected to act as the backend service technology. Amazon Web Services fared in the evaluation through their extensive tutorial matters and pricing model. Learning curve in the case of AWS performed superior compared with other rivals. Their service-managing portal is the clear and easy to comprehend.

Table 2. *Cloud service provider evaluation.*

Feature	AWS	Azure	GCP	Weight factor
Pricing	90	40	50	20 %
Learning curve & Tutorials	95	85	60	20 %
Future Existence	100	100	75	20 %
Security	100	100	100	10 %
Data privacy	95	95	70	20 %
Reliability	80	80	50	10 %
Total scores	**94**	**82**	**66**	**100 %**

3.1.3 Dashboard technology selection

Dashboard operates the frontends technology and visualizes the process data for the user. Dashboard acts as user interface. Low learning curve, pricing, report creation possibility and easy implementation style were the key figures when making the final selection. Positive value was also given in the case where REST interface could be natively supported. Highest weight factors where provided for low learning curve and report creation possibilities. Reason for such action emerged from design rule where Small and Medium sized enterprises were kept in mind for one possible implementation environment. Mentioned business sector could alter the Dashboard solution in-house basis and

create modified reports out from the manufacturing process. Methods in the evaluations were similar compared with the assessment of cloud service provider. Through the Table 3 Wapice IoT-Ticket was selected to be the frontend technology.

Wapice IoT-Ticket managed with the native REST interface and their feature in report creation. Reports can be triggered automatically after the process is finished and report templates can be composed with company specific emblems and document styles. In the case of pricing, the situation was slightly cumbersome. Conventional web applications proved to be the one for least expensive. However, costs would rise in the implementation part. Learning hours would be higher compared to rivals, thus stacking the costs.

Table 3. *Dashboard service provider evaluation*

Feature	IoT Ticket	Freeboard.io	Ignition IIoT	DGLogik5	Conventional Webb Application	Weight Factor
Pricing	60	85	60	60	100	20 %
Low learning curve	85	95	75	70	20	30 %
Report creation	100	10	50	50	100	30 %
Native REST support	100	100	70	70	100	20 %
Total scores	**87.5**	**58**	**51**	**50**	**64**	**100 %**

3.2 Application Layer

Application environment makes the connection of the thesis to production engineering technology and it provides the platform for implementing the designed solution in actuals process. The application is located at the Tampere University of Technology, Mechanical Engineering and Industrial Systems research Laboratory of Laser Applications. Described application is a testing environment for Direct Energy Deposition Methods, a subclass of Additive Manufacturing. Used techniques include cladding and form fabrication. These technologies and methods are detailed in Chapter 2.1. Devices of the application are:

- ABB IRB 4600-40/2.55 robot
- AB IRBP A-750 positioner
- Fronius Cold Metal Transfer (CMT) Advanced 4000R and VR 7000 controller
- Fraunhofer IWS COAXwire
- Corelase C-LASE 3000 kW fibre laser
- Medicoat AG Powder - Duo Laser feeder with FhG COAX8 powder nozzle

These devices are connected through various interfaces, which are noted in the Figure 10 where the structure of the application environment is portrayed. Robot acts as the master in the process, controlling the devices. Robot was the natural choice for the main controller for its capability of handling multiple interface architectures. Especially in the

process cell without any additional programmable controller. During the process, robot adjusts the process variables for the devices. Thus, robot saves the values in text file for later sending to cloud platform through FTP. After the process is finished robot appends the final entries to the file sends the file to the cloud.

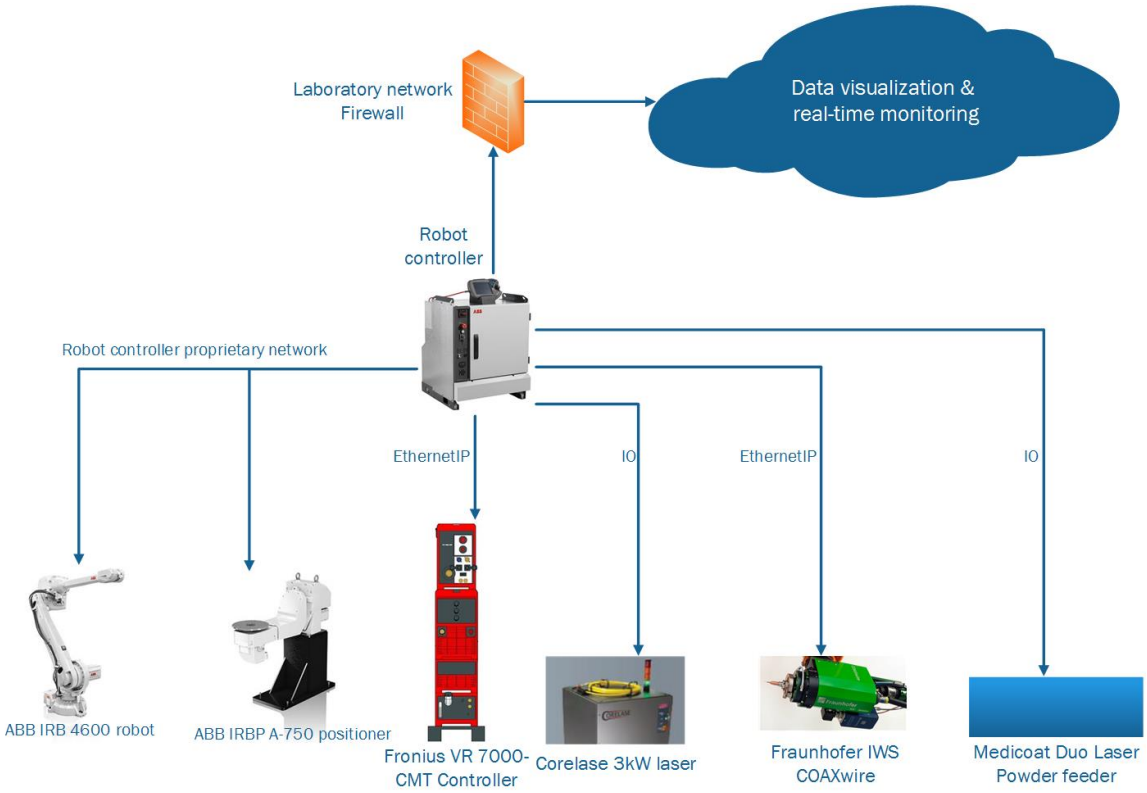


Figure 10. *Application environment overview, adapted from [15; 16; 136-138].*

Fronius CMT device is connected to the robot controller via EthernetIP bus. Robot has own software adapter for accessing the bus and so worth communicating with CMT device. For any third party users CMT device is closed excluding their own Fronius Xplorer software capable of gathering, storing and analysing the process variables from the device and illustrating maintenance diagnostics [139]. Fronius Xplorer is a windows based software without any additional interfaces for external solutions [40], complying with the theory of silos described formerly.

Fibre laser used in the process is a product of Corelase, a Finnish laser manufacturer, and it has the maximum output power of 3000 watts. Connection with the laser and the robot controller is handled with inputs/outputs (IO) interface. COAXwire provides EthernetIP (via Anybus Gateway) for making the connection to the controller, robot in this case. The last actuator unit in the cell is Medicoat powder feeder connected with IO's to robot controller. Together all these devices form the final process cell where research and manufacturing with DED method takes place.

Managing the process is orchestrated either from tailor made robot user interface or by offline programming the robot with ABB RAPID robot programming language. Offline programming is accomplished with the help of ABB RobotStudio (RS) Integrated Development Interface (IDE). Robot UI holds selections for setting up the process hastier when compared to programming conducted with offline or by teaching in online. UI has a navigation menu accessible by ABB FlexPendant Teach Pendant Unit (TPU) touch screen buttons. Through these buttons user can select the process from predefined settings and start the process by simply clicking the *Play* button element. Basic supervisory control is accomplished from FlexPendant TPU. Robot is also capable of form fabrication from Computer Aided Manufacturing (CAM) files. Files must be inserted with G-code language and robot has a custom-programmed feature for altering the G-code path file to RAPID programming syntax. Selection of the CAM designs are handled from UI as well.

After the user has placed the building platform on the positioner (see Chapter 2.1.1 for details) and made the required selection via UI or custom offline programming with RS, robot can start the form fabrication. Robot picks the desired tool (COAXwire or Powder Feeder Nozzle or CMT torch) from the tool changer and produces the article. Used tool can also alter during the process. In this case, robot changes the tool via tool changer. During the production, robot saves the process data inside the text file with desired frequency. Furthermore, during the process, the real-time process monitoring is conducted via robot REST interface.

ABB Robot's REST services are described in textual format at ABB robot developer's web pages [41]. Service uses the Digest authentication with cookie management and the data can be requested either with XML or JSON format, as the default data is responded in XML. Data is requested as JSON, with the query of *json=1*. Through the REST interface robot's variables and IO's can be modified and monitored. Path for the variables inside the robot controller are portrayed with tree structure at the developer's page. [41] Yet, the developer needs to have the knowledge over the robot program structure to perform the right requests for the right variables. Mentioned tree structure is illustrated in the Figure 11 and it corresponds to the structure of the robot program in ABB RS IDE.

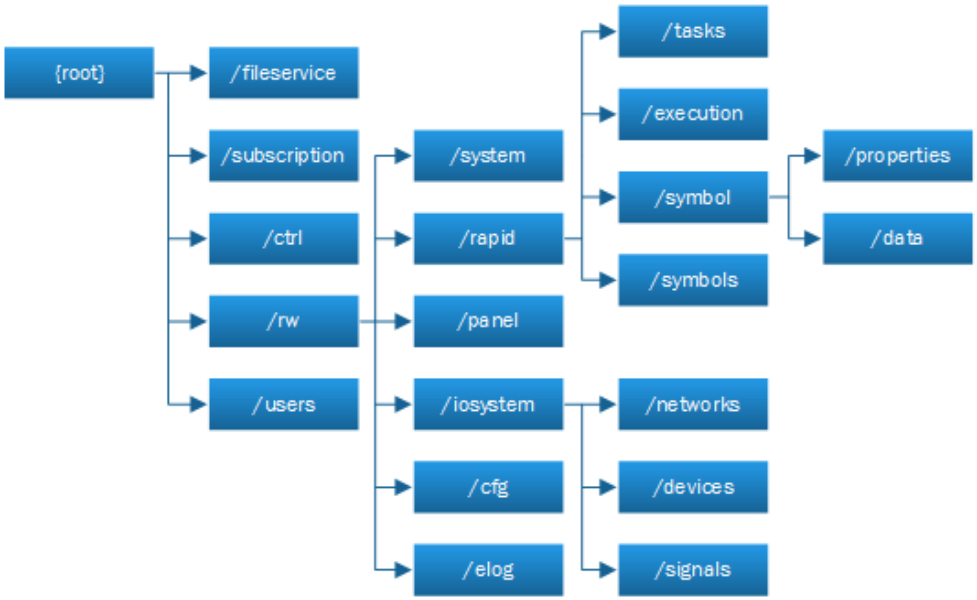


Figure 11. *ABB Robot RESTful description's tree structure, adapted from [41].*

According to the REST services design guidelines the resources should be represented with nouns rather than verbs [46]. This design rule is adopted in the Robot REST services as it can be noted in the following example, where RobotStudio virtual controller (localhost) is accessed and the state of the variable *reg1* is requested in JSON format. Similar action can be performed to actual robot controller, in such case the localhost is replaced with controller's IP address.

http://localhost/rw/rapid/symbol/data/RAPID/T_ROB1/user/reg1?json=1 (1)

3.3 Backend technology

Amazon Web Services was selected to acts as the backend server technology in the implementation. Through AWS one-year Free Tier offer, multiple AWS services can be studied and incorporated in the backend solution. This Free Tier method proved to be quite convenient for executing the program code and when storing the data in AWS cloud in both unstructured textual and structured relational database format. Following section describes the AWS cloud ecosystem as the whole and then portrays the individual services used in the implementation. Reasons for implementing named services are additionally detailed.

3.3.1 Amazon Web Services ecosystem

Both [115] and [31] state that AWS serves as IaaS paradigm. However, this is not the whole story for there is an alternative categorization. Some AWS services such as EC2 act as IaaS in their plain form. Situation alternates when one of the available instances are deployed at the service. From that point, onwards the EC2 can be categorized as

PaaS. Another aspect arises in the case of services providing platforms for application execution. Elastic Beanstalk operates on this field and can be at first classified as PaaS. AWS Simple Storage System on the other hand acts as SaaS basis. Unstructured data can be stored and retrieved in the service with API's and stored data can be inspected via AWS portal.

For monitoring and assessment purposes, AWS customers possess multitude of means. In the implementation phase, some of these were found rather useful. One of these is the access to AWS Dashboard (AWS portal web page) from which the deployed resources can be monitored. Dashboard illustrates the charges for current month and indicates whenever the Free Tier offer has been exceeded. From the AWS portal users are able to found a *Budget Tool*. A feature that comes in profit for constraining the monthly costs for example in the training or familiarizing phase. Notification can be triggered on the forecasted budget or actual budget exceeding. Amazon CloudWatch is a tool for monitoring resources state and usage with graphical overview [140]. Free Tier offers 3 CloudWatch Dashboards with 10 custom metrics and alarms [64].

AWS has designed the hands-on work with the services to be quite straightforward with low learning curve. Adopting new services and making the adjustments with the existing ones is handled via AWS portal. Only after few hours of learning, the basic usage of the services and configurations comes clear. Within the same timeline user can get a good overview from where more detailed options are found in the portal.

3.3.2 Amazon Virtual Private Cloud

AWS provides a concept Virtual Private Cloud or usually addressed as VPC [141]. It is a private virtual network inside Amazon cloud ecosystem similar to customer's private network inside the company premises. Various AWS resources and services can be deployed inside the VPC network and customer can have multiple VPC networks at their disposal without any additional charge. Each of these networks are logically isolated from any other AWS virtual networks. [141] Virtual Private Clouds can hold different subnets, which furthermore can be dedicated to either private or public use, latter for connection to internet. [142] Subnets must be located inside one Allocation Zone in one Region (for an instance at EU region, Frankfurt Allocation Zone). This gives the customer the availability for redundant operations. On the creation of the VPC user is requested to specify the range of IP addresses to be used. These IP addresses are given in Classless Inter-Domain Routing blocks (CIDR) blocks. [143] In the Figure 12 there is an illustration of the VPC concept where VPC has CIDR block 10.0.0.0/16. Meaning that first 16 bits in the address are reserved for the network part and remaining 16 bits for subnet allocations [144; 145]. Subnets can be assigned into this particular VPC and a subnet must be a subset of the VPC CIDR block. For assigning the subnets, AWS recommends to use the addresses from non-public routable range according specification

RFC 1918 (Request for Comments). Additionally, some of the IP addresses are reserved for AWS proprietary use. [143]

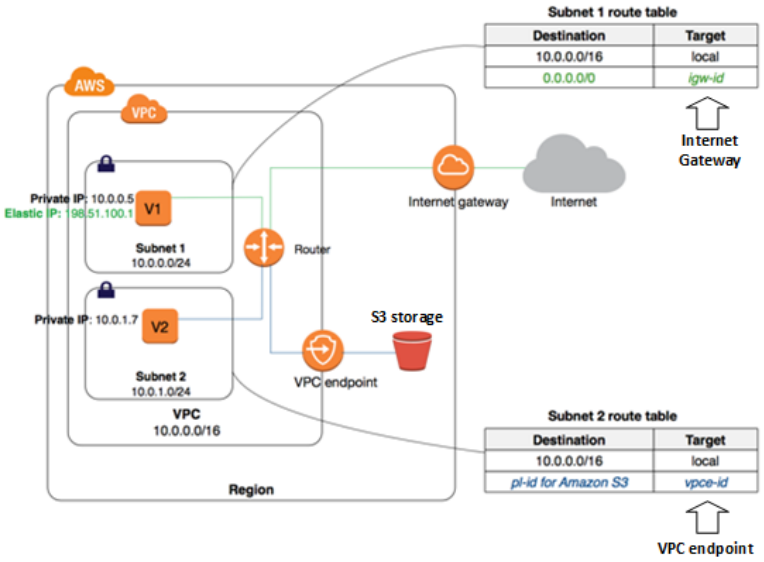


Figure 12. *Amazon Web Services Virtual Private Cloud, adapted from [142].*

Every subnet must be attached to AWS routing table. Routing table specifies the rules for outbound/inbound traffic [143]. In the Figure 12 subnet 1 possess a running instance marked V1 and routing table has been configured for accessing AWS Internet Gateway (IGW) (igw-id) [146]. IGW is a VPC component for making the internet access conceivable from inside the VPC network and making the Network Address Translation (NAT) for instances with additional public IP addresses (see Elastic IP later). Purpose of the Internet Gateway is to alloy the internet outbound/inbound traffic only with the instances connected to IGW through router. Method, which enables traffic within the VPC between different subnets and only dedicated subnet for accessing the public network. [146] Example in Figure 12 has an Elastic IP address (Static IP) assigned for instance V1. Meaning that according instance can be found from internet with this IP address prior to any change from the user itself. [147] After altering the IGW and so worth according routing table AWS relays the traffic for appropriate instance [143; 146]. AWS provides one Elastic IP address for one customer account. Any additional Elastic IP's are charged. This is a consequence of the limited amount of AWS disposable static public IP's. [147]

In the Figure 12 subnet 2 is not allowed for internet access yet it is connected to a concept called VPC endpoint via routing table. VPC endpoints are AWS virtual devices alloying the traffic from AWS subnet for additional AWS resources without public internet connection, NAT device, VPN connection or AWS Direct Connect. [142] Thus, the traffic never exits the AWS dedicated network. In according figure, subnet 2 has a routing table configured for relaying AWS Simple Storage Service (S3) traffic through VPC endpoint (vpce-id). VPC endpoint configuration takes place in AWS portal with

indications of the prefix ID (identification) of the service. ID is formed according to AWS region and service name; *com.amazonaws.<region>.<service> [142], com.amazonaws.eu-central-1.s3.*

For the security purposes AWS offers three distinct features, Security Groups, network Access Control Lists (ACL) and Flow logs. Security group is controlling the inbound and outbound traffic for the instance connected to internet. ACL technology manages the inbound and outbound traffic between different subnets. AWS recommends ACL commissioning for additional layer of security while stating that security groups are sufficient to provide necessity protection. Flow logs capture the information over the traffic between different interfaces inside VPC. [148] In the implementation Security groups where the main layer of security protection. Security group functionalities are detailed in Chapter 3.3.3.

3.3.3 Amazon Elastic Compute Cloud

Customer's using AWS cloud platform are eligible to launch AWS Elastic Beanstalk service. A platform for executing custom web application or other custom program. Elastic Beanstalk has a support for applications written in multitude of programming languages; Java, .NET, Python, Ruby, Node.js, Go and Docker are available [149]. Elastic Beanstalk has great advantages from easy deployment of code to rapid initialization at first launch. Regardless of these assets, making large-scale application AWS Elastic Compute Cloud (EC2) is a better service. For the implementation, there was an initial study for the possibility to deploy application on Elastic Beanstalk. Limitations in the features of questioned service induced the movement for EC2 service.

Amazon EC2 operates on IaaS basis. Customer can select an instance to be launched from AWS ready-made packages called Amazon Machine Image (AMI) or create their own image which contains customer proprietary applications, libraries, configurations and data. [150] Selection and customization takes place in AWS portal. Prior to launching an EC2 instance, a VPC should be configured. Launch of an EC2 instance proposes to use a default VPC (created at the AWS sign-up) and deploy the EC2 inside. For making future modifications adaptable, setting up a custom VPC and proper subnets is highly recommended.

When launching new EC2 instance user is requested to choose the AMI (ready-made or custom). Conceivable instances consist of 31 variations of Linux and Windows OS images from which 15 are eligible via AWS Free Tier. Against Linux images, one of the most popular version is *Amazon Linux* including default command line tools of Python, Ruby, Perl and Java with additional repository packages of Docker, PHP, MySQL and PostgreSQL. AWS Free Tier offers 750 hours of monthly execution time for selective instances [64]. These instances have furthermore certain limitations [151]. Instance type

has to be *t2.micro* which incorporates 1 CPU with clock speed of 3.3 GHz and 1 GB of memory. Disk space is within the limitations of AWS Elastic Block Storage (EBS) Free Tier offer [151]. EBS disk space is limited for 30GB in Free Tier [64].

Security groups are one network breach-evading concept in AWS services. When launching an EC2 service, desired security group is assigned by the user. If no security group is allocated, AWS uses the default settings allowing all traffic in both directions. There are few additional notes for setting up the security rules. Rules are always permissive so no rule can be created for denying the access. There is a short delay from setting up a new rule for AWS taking the rule in use. Security groups act as stateful manner. When request is sent for users instance the response messages are allowed regardless of the security rules. [152] Each inbound and outbound rule is consisting of following settings [152]:

- Protocol type to allow (e.g. HTTP, HTTPS, FTP)
- Protocol to allow (TCP / UDP)
- Port number to allow
- Source IP address or address range to allow

When launching a Linux instance customer is granted with *.pem* file for holding the key pairs making the SSH connection with the running instance. Opening the connection can be handled through SSH client such as PuTTY [153]. PuTTY is an open source and so worth entirely free [154] which has induced the popularity of the software. PuTTY utilizes *.ppk* security key files causing the need for transform from *.pem* to *.ppk*. Such action can be performed with PuTTYgen software [153]. After the transformation, *.ppk* file, EC2 instance network address and used protocol can be inserted in PuTTY software making the terminal window connection [153]. For file transmission purposes FTP client software should be utilized. Similar settings are conducted prior making the connections [153].

After initiating the connection with the EC2 instance, the required software's can be installed and runtime configurations can be performed. Custom applications are then deployed and tested. Following the desired configurations and application deployments made, customer can register their own AMI. Feature, which comes convenient when similar instances has to be launched, or backing up the configurations and software's for specific instances. Registered AMI's can be later deregistered. According feature is portrayed in Figure 13. Customer instances are stored in AWS EBS [155]. For Free Tier usage maximum of 30 GB of image disk space should be considered [64].

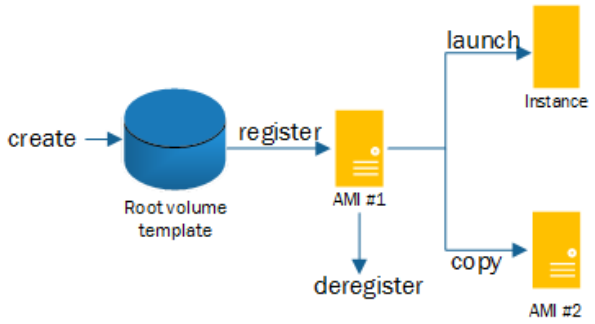


Figure 13. *AWS custom AMI creation, adapted from [155].*

3.3.4 Amazon Simple Storage Service

Various applications and solution often have a need for storing data in its native format. Amazon Web Services Simple Storage Service solves this introduced dilemma. In the implementation S3 bucket is established for storing process data files in their native textual format. These files can be later commissioned for divergent usage or to be moved for other file storing platform. Modularity of the solution and centralizing the original data in one location are the key features for utilizing S3. Establishing S3 storage takes place via AWS portal. S3 storages are called buckets in AWS ecosystem [156]. Customer can operate with multiple buckets, inside different regions and save identical data in multitude of these buckets for making redundancy capable features. AWS itself stores the data in numerous servers and multiple facilities within selected Availability Zone. AWS serves the customer with checksum calculation for all the datagrams inside their network. Mismatch in checksum can be indicated and corrupted data can be repaired. AWS utilizes their own self-healing method for corruption correction, thus improving data integrity. [156] AWS Free Tier enables 5 GB of disk space for S3 buckets with 20 000 Get requests and 2000 Put requests per month [64].

S3 stores the data as objects. These objects can be written, read or deleted whether user has the subsequent permissions. Customer can control the access permissions for the bucket and in addition grant the access only from specific locations around the globe. VPC endpoints are implemented for creating a secure connection when the data is moved within the AWS ecosystem. In data transition from end terminal (workstations) into S3 bucket, AWS utilizes SSL encryption. This action is transparent for the user thus, requiring no additional steps. [156] Data transmission is enabled with two distinct methods. First, customer can use web page based AWS portal with simple “drag’n drop” feature. Second, customer is eligible to operate with AWS Software Development Kit (SDK) [156]. SDK’s provide application level access for files in various buckets. AWS SDK has the support for Java, PHP, .NET, Python, Node.js and Ruby programming languages [156]. In Node.js language AWS SDK is accessible via node package manager and S3 Node.js API is available at AWS web page [157].

Access keys are used when opening the connection from Node.js environment in specific S3 bucket [156]. These access keys can be generated at AWS IAM portal. Access key consists of *Access Key ID* and *Secret Access Key* pair. [158] Using these keys together with the region identifier and bucket name, a connection to the specified bucket can be opened. IAM keys are monitored and customer can track the usage of all the generated keys from AWS portal [158]. In case where keys are applied in different context than originally designed, customer may consider a security breach.

3.3.5 Amazon Relational Database Service

AWS S3 is a service for storing files (objects) in unstructured manner and in their native format. When multiple applications are accessing the same data or when extensive amount of data should be employed, databases are the right tool [159]. Relational database model is a development from flat databases. In relational database, data is stored inside tables, which are related with each other through links. This enables the multilayer ideology when storing the data. Methodology in relational database is based on theoretical background of predicate logic, relational calculus and set theory. [159] Process history data in the implementation is in plain string entries in textual file. This format can easily be structured in database tables. For future analysing purposes, the gathered data is stored in AWS Relational Database Service (RDS). Multitude of the AWS technology partners has the adapters and knowledge for accessing their customers RDS databases. A great opportunity for future analysing of the data with any third party technology. Learning from usage of AWS QuickSight is additionally conceivable when data is stored in AWS RDS.

RDS offers 6 different database (DB) engines from which the customer can make the selection. Amazon Aurora, MySQL, Oracle, MariaDB, Microsoft SQL Server and PostgreSQL are represented.[160] AWS RDS services are executed within additional AWS instance. Performance of the instance can be selected from various options yet when applying AWS Free Tier, *t2.micro* instance is the only conceivable one [64]. According instance is provided with 750 hours of monthly execution time. Sizing is limited for 20 GB of database storage inside general purpose SSD (Solid State Drive) disk [64]. Of course, detailed limitations are not eligible when Free Tier is expired or not used. MySQL database engine was selected for the previous knowledge over the according database and the experience of the MySQL database engine adapter in Node.js environment.

When deploying a new DB, customer can select the time interval for backup purposes. AWS automatically takes an image (dump) from the database for later restoring. This action uses EBS capacity, which on the other hand reduces the capability for storing custom images of EC2 instances. When recalling the Free Tier offer for 30 GB EBS storage and 20 GB of DB storage, customer should be aware of the possibility for cross-

ing of the Free Tier limits. Actual connection for the database is handled via existing EC2 instance. Database can be perceived to be a specific data storing location for certain EC2. Ideology is portrayed in the Figure 14.

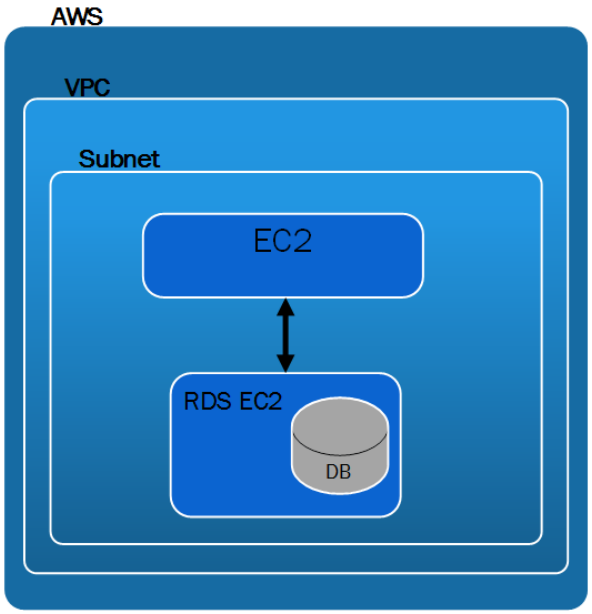


Figure 14. *AWS RDS Database deployment ideology.*

API connection with the database orchestrated over MySQL package for Node.js. Configuring of the *username*, *password*, *port* and *database name* are moreover required [161]. For visual inspection of the database, MySQL Workbench [162] can be harnessed. Workbench is conceivable from MySQL Community Edition package without any charges [162]. Connection from Workbench into MySQL database is configured similar with the API connection. Merely the connected EC2 address is furthermore required. Adapter for making the connection from Workbench into RDS is quite tardy with Free Tier low-level instance and for extensive use, higher level RDS instance should be considered.

RDS with MySQL database engine operates on SQL (Structured Query Language) method. A standardized querying language for relational databases. With SQL, applications and users are eligible for correlating and manipulating the data between relational tables. SQL syntax holds various commands and combination of these commands can be created. Basic statements are SELECT, INSERT, UPDATE and DELETE. SELECT is being used for reaching certain data from relational database. INSERT is deployed when new data needs to be inserted into the database. UPDATE amends the data inside database. DELETE eliminates subjected data from the database. [159] W3Schools [163] holds a complete list over the possible queries.

3.4 Frontend technology

IoT-Ticket was selected to serve as the platform for Dashboard functions. For making the visualizations according platform has functionalities on monitoring, controlling and reporting creation compared with optional service providers (see Table 3 for details). With IoT-Ticket, customers are capable of making Dashboards on gathered data with minimal effort. IoT-Ticket tools are entirely web browser based; no software's or frameworks are required to be installed on customer's servers and workstations. This liberates the resources for other purposes and unties the attachment on working with certain workstation having the right tools installed. In the following, IoT-Ticket platform is represented.

3.4.1 IoT-Ticket Platform

IoT-Ticket Platform as a whole can be perceived to be a set of various devices, finally brought together with IoT-Ticket Dashboard. These devices and platform concepts are portrayed in Figure 15. In monitoring solutions, different devices and actuators are connected into IoT Device inside the manufacturing company. IoT Device can be Wapice proprietary WRM247+ (Wapice Remote Management), customer's own device, some third party device or server operated application supplied with suitable interface (see Figure 16 for details). Via secured connection, data is transposed into IoT-Ticket Data Server. From the Data Server variables can be adapted into Dashboard UI operated over secured internet connection and illustrated at the end device. IoT-Ticket Platform is additionally capable of bidirectional communication. Machines and devices can be controlled from the Dashboard solution as detailed in Figure 15. [117]

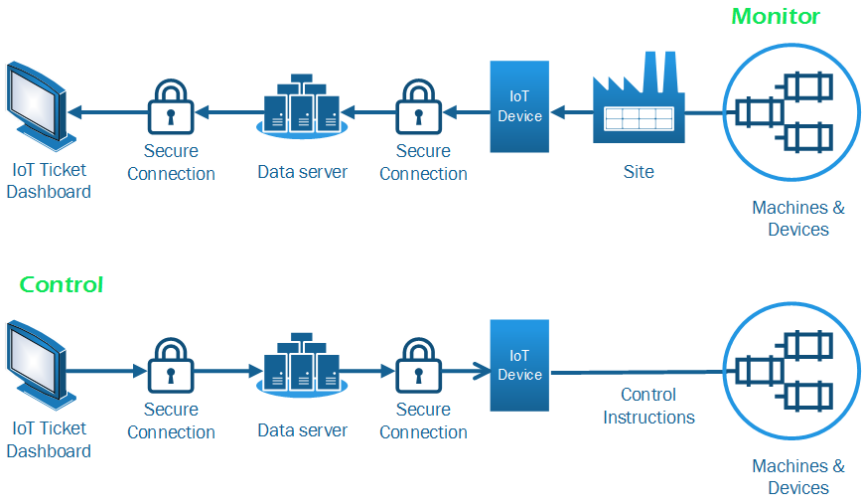


Figure 15. *IoT-Ticket Monitoring and Controlling methodology, adapted from [117]*

Advantage with the IoT-Ticket lies in its multitude of characteristics. IoT-Ticket can operate either PaaS or SaaS basis. Customer can purchase only the platform and contribute internally for the solution development. However, Wapice offers services to build specific solutions for customer's needs. IoT-Ticket Platform can serve the whole chain from hardware devices to software services for creating customer's solution. This reduces the need for surveys and studies performed by the companies, in the case where countless parties contribute. [117]

IoT-Ticket supports multitude of interfaces. One of these interfaces is WRM247+ device. WRM247+ is a Wapice proprietary robust device performing remote measuring, controlling and management for the actuators or events at the shop floor level. Questioned device can support 10 different interface modules or protocols. Acceleration measurement, CANdata (Control Area Network), CANopen, One-Wire, ISO11783 Monitor, Modbus, ModbusTCP, Modbus Server, Time and ODB (On-board Diagnostics). When combined with additional interfaces illustrated in the Figure 16. IoT-Ticket can support totally 18 different protocols or modules. [117]. IoT-Ticket Platform Data Server can natively support OPC UA (OLE for Process Control Unified Architecture) protocol. Such case when customer merely possess Classic OPC interface an OPC Gateway can be commissioned. [117] Newest interface for IoT-Ticket Data Server is Libelium. Libelium provides wireless devices with low power, long range (21 km) capabilities for smart city and smart environment applications. Now these devices are eligible to be connected with IoT-Ticket Data Server using the advantages of the platforms Dashboard and Reporting tools. [164] IoT Tracker is a mobile phone based interface making user's end device an IoT device. With Tracker interface mobile devices sensor data is available for IoT-Ticket solutions. Wapice provides REST API libraries for Java, C# and C/C++, Python, Codesys, Qt and Simulink programming languages. REST API Development Guide is additionally available for using another programming language,

connecting directly over REST interface. [117] Practical part utilizes JavaScript programming meaning that mentioned API Development Guide was studied.

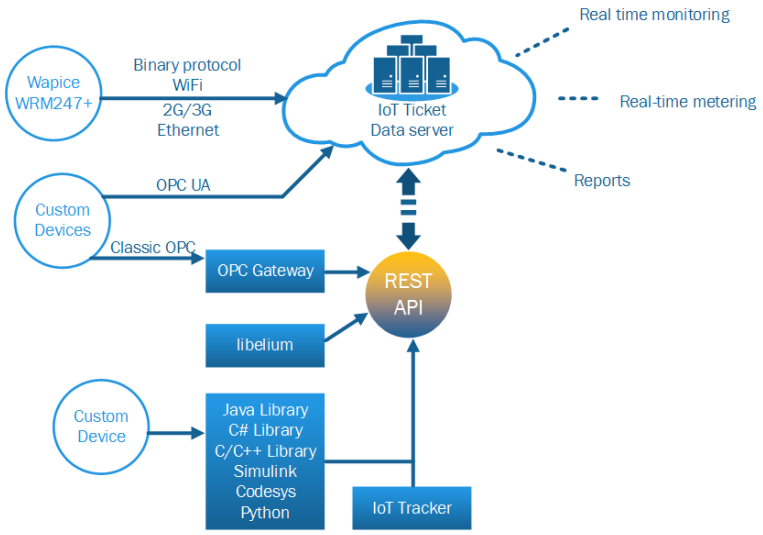


Figure 16. *IoT-Ticket Connectivity diagram, adapted from [117]*

IoT-Ticket has a hierarchical data model initiating from root level named Enterprise (e.g. customer or company). Enterprise can have multiple devices and these devices can have multiple datanodes (datatags). Inserted data is written or read, from these individual datanodes. [165] Figure 17 portrays a situation where example enterprise has a vessel. Vessel acts as device and according vessel has path *Engine/LeftBlock* for defining the datanodes *Temperature* and *AirIntake*. Additionally, vessel global positioning can be reached with two distinct devices.

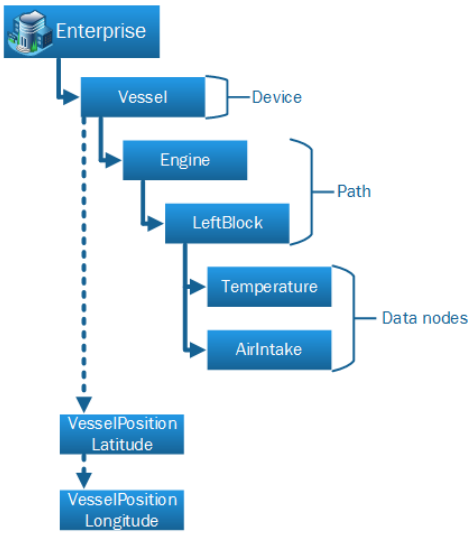


Figure 17. *IoT-Ticket Data Model, adapted from [165]*

IoT-Ticket API Server has a quota policy for users taking an advantage of the 30-day evaluation period. With this, IoT-Ticket platform is restricted to 5 devices. Additionally,

these devices can have 20 data nodes per device and 20 000 read requests per device in one day. When using a commercial version of IoT-Ticket platform quota size is defined according the subscription level. [166]

Requests for IoT-Ticket Server are made with HTTPS. Thus, protection against wire-tapping and man-in-the-middle attacks can be reached. Authentication of the requests are handled with HTTP basic authentication. Username and password need to be provided each time the when message requires an authentication. [165] Authentication requirements for the distinct messages can be noted from the Table 4, detailed in the next paragraph. In practise, every message needs to be authenticated. When client is registration a new device, under the Enterprise root element, an ID is provided for later accessing according device. [165] Registration and data writing sequences are illustrated in the Figure 18. First client sends the request for device registration. IoT-Ticket API Server creates a new device and responses with device ID. After the client has stored the provided ID, client is applicable to start sending data to device datanodes. If IoT-Ticket API Server notices that datanode does not yet exist, one is created and data is stored. Client receives a response message for succession or failure of the writing procedure.

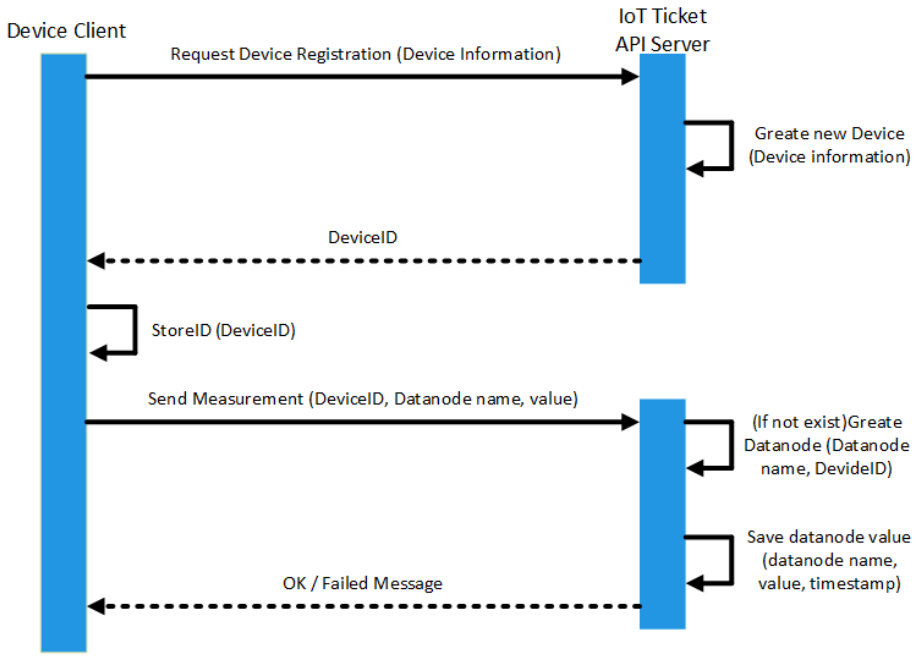


Figure 18. *IoT-Ticket Device registration and data writing sequence, adapted from [165].*

With IoT-Ticket API Server REST interface, specific URL (Uniform Resource Locator) and HTTP methods define the action to be performed at the server side. Total 8 different resources are available. [165] These resources are detailed in Table 4. For space saving purposes table represent only the last section of the URL. Initial part of the URL is consisting of base URL, name of the API and version number. Following URL gives an example where device is being registered:

https://{base-ul}/api/v{version-number}/devices (2)

In the above sample URL: *{base-ul}/api/v{version-number}* can be shortened for: *{api-server-URL}*. This comes convenient later in the Appendix A where request-response message airs are detailed.

Table 4. *IoT-Ticket API Server resources [165]*

Operation	**Purpose**	**URL**	**HTTP method**	**Authentication required**
Register a device	Create a new device under Enterprise	/devices/	POST	Yes
Get devices	Request for clients available devices	/devices	GET	Yes
Get a device	Request information over the device	/devices/deviceID	GET	Yes
Get device data node list	Request the list of device data nodes	/devices/deviceID/datanodes	GET	Yes
Write data	Write the vales to data nodes	/process/write/deviceID	POST	Yes
Read data	Read device data node values	/process/read/deviceID	GET	Yes
Get client quota	Request a client quota information	/quota/all	GET	Yes
Get device quota	Request quota information for certain device	/quota/deviceID	GET	Yes

In the HTTPS messages, communicating with IoT-Ticket Server, payload transposes the username, password and the required information, proprietary for each of the resource. Clearer picture for the usage of the resources can be given through series of tables. These tables are represented in Appendix A. Tables illustrate the *request-response* message pairs and the structure for the most essential resources given in Table 4. Response can be requested in XML thus JSON format is the default. [165] JSON format was used in the thesis implementation.

Error handling with IoT-Ticket API Server takes place with HTTP status codes and additional information in the body of the error message. Table 5 indicates the possible status codes for the responses and the Table 6 indicates the structure of the body in the error response message. Final table in the set of error handling, Table 7, describes the API Server specific error codes.

Table 5. *IoT-Ticket API Server HTTP status codes [165]*

Status code	Note
200	OK
201	Created
400	Bad Request
401	Unauthorized
403	Forbidden
500	Internal Server Error

Table 6. *IoT-Ticket API Server error message body description [165]*

Response message body	
Field	**Description**
description (String:500)	General information over the error
code (int)	Internal error codes
moreInfo (String:255)	Reference to the documentation where additional description can be found
apiver (int)	The API version

Table 7. *IoT-Ticket API Server internal error codes [165]*

Code	Description
8000	Internal server error
8001	Permission is insufficient
8002	Quota violation
8003	Bad parameters
8004	Writing failed

3.4.2 IoT-Ticket Dashboard

IoT-Ticket Dashboard is a web browser based and secured interface for monitoring, controlling, managing the devices. Visualization are created with online-based content creation tool. Customers can exploit multiple Dashboard pages starting from the supervisory level of the application and going deeper inside the sites and down at the individual devices. IoT-Ticket Dashboard has been divided in two different tools for creating the final visualization, *Interface Designer* and *Dataflow Editor*. [117]

Interface Designer is a tool for Dashboard creation. A user can utilize "drag n' drop" mechanism for adding distinct elements of labels, gauges, charts, maps, camera views, sliders, buttons and indicators. Elements are moved on suitable position on the canvas with user experience kept in mind. When adding a new element, data variables are attached with this specific element by employing "drag n' drop" method from the Datatags (datanodes) list. These datatags are the variables inserted with selected interface into IoT-Ticket Data Server. [117] Interface elements can further be configured for process specific minimum / maximum values, scale of the measured variable and units of variable. Yet, unit information can be inserted together with variable data over REST interface. User can be provided with button elements acting as links between different Dashboards and Dashboard pages. Otherwise, user is required to open the certain Dashboard and further select the according page.

Second tool for Dashboard formation is the Dataflow Editor. Dataflow Editor is an IEC 61131-3 inspired block programming editor operated via web browser. Different Dataflow Editor widgets are connected with each other by elastic wire method. Interface Designer and Dataflow Editor integrate seamlessly together creating the final Dashboard visualization. Dataflow Editor provides function blocks/widgets for creating complex logical operations used for executing control actions at the shop floor level. Formed logical operations can otherwise be used for triggering visualization elements for providing monitoring action. [117] Dataflow Editor is also used for manipulating the datanode values before representing those to the user. Calculations can be made with selecting right function blocks. Calculations include basic mathematical operations with some additional ones.

3.4.3 IoT-Ticket Reporting

IoT-Ticket Reporting tool is based on the similar features compared with the Dashboard solutions. With Report creation tool user can "drag n' drop" elements on the document template and create the visualizations for the report. However, the selection of the widgets are restricted. Reports cannot represent any communicable elements. Elements can be static (e.g. company logos) or dynamic. When using dynamic elements, Datatags are inserted with "drag n' drop" method. After the report template is created, customers

can use the same template and trigger the automatic creation of the report updated with the dynamic elements. Triggering can take place from various sources. Application level process finishing could be one of these triggering actions. However, more complex logic can be utilized. Reports can incorporate charts, sliders, text boxes, figures, dialogs, checkboxes, labels, gauges, camera snapshots, maps, and state indicators. Reports can be delivered at users emails and different templates can be used for different departments inside the company. As an example, material processing team can receive different report with different elements compared with automation team receiving information with their interests. [117]

Report creation tool differs from Dashboard editing tool set for holding one adjunct browser tab. This adjunct tab is used for previewing the outlook of the report under design. Otherwise, report creation takes place with Report editor and Dataflow editor. When initializing a new report, administrator should take into consideration the levels of the datanodes and user experience for accessing the finalized report. Reports can hold any of the datanodes regardless of the level where the report was first initiated. However, the end user can work with more intuitive manner when report is located under the path, dependent for the content of the report. As mentioned, reports can be triggered with predefined timetable or with trigger signal from any of the Dashboads.

4. IMPLEMENTATION

Following chapter describes the implementation where cloud platforms are accessed for creating process data monitoring solution. Solution is developed for application environment, described in Chapter 3.2. As mentioned, the application environment was installed in the laboratory with all the basic hardware and additional devices before starting the thesis. For this reason, no affect from the thesis were made for immediate alternation of the environment. Multitude of future prospects for updates where nevertheless considered and during the thesis project, intensive co-operation was conducted for setting up the final application. Consequently, subsequent chapter concentrates on software development of the cloud platform working tightly together with the hardware level.

For providing a clear presentation for the viewer, following subsections are divided in five distinct parts. Chapter 4.1 gives the viewer an overall representation over the used dataflow and security methods. Purpose of this chapter is to bring forth more clear conception over the implementation. Chapter 4.2 describes the cloud platform framework used as backend for the implementation. For the frontend solution, three specific entireties can be portrayed. Thus, following chapter is divided in additional three parts. Chapter 4.3 covers handling of the real-time process monitoring and illustration of the process variables for the users. Chapter 4.4 concerns gathering of the process data simultaneously with the active process and after the process is finished, provide the data for cloud frameworks for later observation. Chapter 4.5 describes the report creation steps for the process.

4.1 Dataflow and security architecture

In the Chapter 3.2 application layer was portrayed. Within this chapter, Figure 10 portrays the mechanical units and actuator devices of the application layer. These devices are crucial for the usage of the environment. However, for the purpose of the thesis, even more crucial is the illustration of the dataflow and accessed security methods within both the application layer and cloud platforms. Mentioned Figure 10 can be modified for Figure 19 to represent these details.

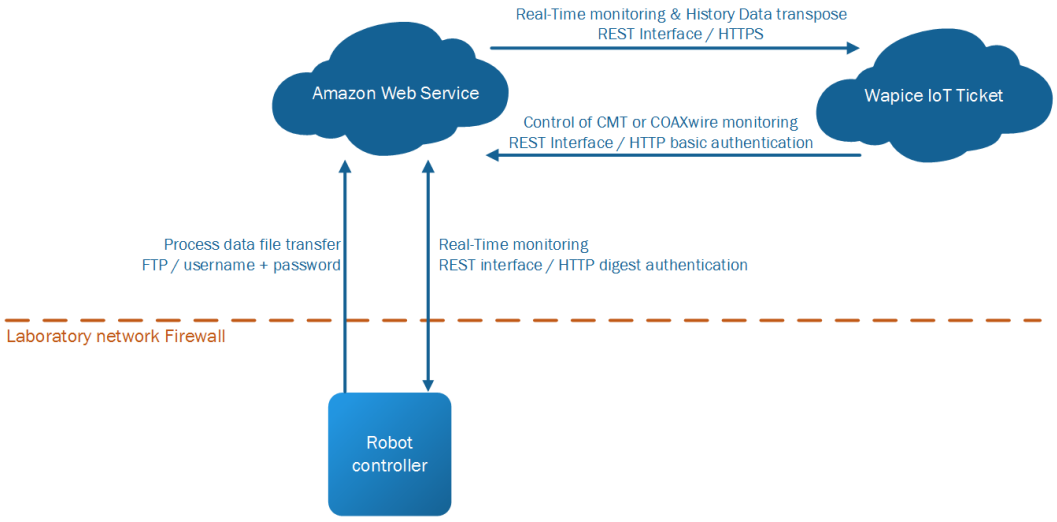


Figure 19. *Implementation Dataflow and security architecture*

Each data transfer uses different security method. For process data transmission, a basic FTP with username and password authentication was utilized. Set against for robot REST API interface, which holds HTTP digest authentication method. Digest authentication holds challenge-response scheme [167]. Scheme challenges to use a nonce value. Response holds MD5 128-bit algorithm hashing for checksum of the username, password, nonce value, HTTP method and requested URI. [167] In the robot Digest authentication also provides a cookie for the client after a successful authentication. Later accessing the same API questioned cookie is transposed with the message to verify the user [41]. Wapice IoT-Ticket holds HTTPS protocol for data transmission and controlling the requested process monitoring, takes place with REST interface together with HTTP basic authentication method. HTTP basic authentication holds a method where user must authenticate itself against the realm by representing username and password [168]. These credentials are hashed with base64 encoding. Basic authentication does not support nor possess cookie mechanism [168].

In the application research project, robot can have multitude of users. Full access rights for the robot features, by all the users, was reduced. Robot holds a default credentials for accessing and usage [169]. These default credentials originally possess a full access to robot features. New user and credentials can be formatted via RS User Authentication Service (UAS) [169]. Default credentials access was lowered to hold only the basic robot operations and new user with new credential where formed to have full access to robot usage and configuration. These credentials are moreover used with REST API communication and by the main users of the cell.

4.2 Cloud platform framework

According to study conducted in theoretical background Amazon Web Services was selected to act as the backend service and provide the resources for the implementation. Various AWS services where harnessed for building up the framework. These services and resources are detailed in Chapter 3.3. Thus, in the following the framework is described as it is set up for the implementation. The most convenient method portraying the framework is to view the architecture through a figure.

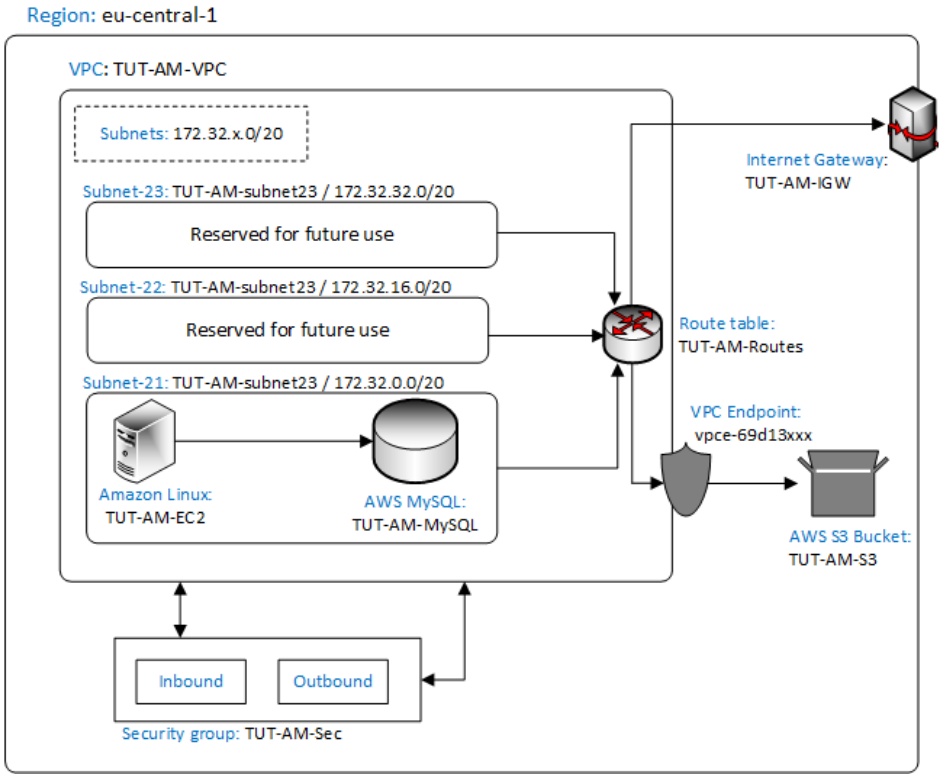


Figure 20. *Cloud platform framework*

At the initial phase, an AWS region was decided for operating all the resources. According to Amazon Customer Agreement [66] data is always kept in the Availability Zone where the user makes the selection. As operating in EU economical region a Frankfurt Germany based *eu-central-1* was selected for the datacentre and resource location. According functionality can be noted from the Figure 20 where region is portrayed as the outer shell covering all other resources. Inside the region, a Virtual Private Cloud was configured. Prior for launching any actual resources inside the VPC a subnet allocation was configured. In the implementation, architecture was divided in three different subnets for leaving two as for future reservation already at this stage. These subnets can later on hold new AMI's for analysing of the process data or controlling the process. Subnet allocation was selected from 172.32.0.0 address space with CIDR block of 20 (172.32.x.0/20) making totally 16 different subnets available with each subnet having 4094 possible IP addresses [170]. AWS route table was configured for allowing traffic

only from the *subnet 21* to be transferred into Internet Gateway (IGW). With this action, only AWS EC2 located inside the *subnet 21* can access the internet. In the future, other two subnet AMI's can prepare the results for *subnet 21* AMI. Inside the route table, another configuration was set forth for allowing *subnet 21* to communicate with the AWS VPC Endpoint. VPC Endpoint was configured for allowing the communication with AWS S3 bucket. As mentioned in the methodology, according configuration is conducted with S3 bucket ID and prefix relating to AWS region. Thus, in the implementation VPC Endpoint was configured as *xx-xxx54007 (com.amazonaws.eu-central-1.s3).*

AWS security group holds the main layer of security for the internet traffic. Security group is illustrated in the Figure 20 as an external service block although this is done only for representation purposes. Security group acts in conjunction with all the services always confirming the inbound and outbound traffic. In the implementation, multiple security rules were put forward for inbound traffic. Totally five different types of communication where needed HTTP, HTTPS, SSH, FTP (Custom TCP Rule) and MySQL with various modification for acceptable IP addresses. Detailed specification of the inbound rules are noted in Table 8. For testing purposes, outbound rules were left in the default settings allowing traffic to all IP addresses.

Table 8. *AWS security group inbound rule settings*

Type	Protocol	Port Range	Source IP	Function
HTTP	TCP	80	Robot IP, used workstations IP's	Robot REST interface
HTTPS	TCP	443	all IP's	IoT-Ticket, secured connection
SSH	TCP	22	used workstations IP's	Terminal connection with EC2 instance
Custom TCP Rule	TCP	20-21	Robot IP, used workstations IP's	FTP server
Custom TCP Rule	TCP	xxxx (secured)	all IP's	used for inbound communication from IoT-Ticket, basic HTTP authentication
AWS RDS MySQL/Aurora	TCP	3306	all IP's	route table handles the accessible communication

Inside the *subnet 21* AWS Elastic Compute Cloud with AMI of Amazon Linux was launched. Later on this modified Amazon Linux was packed as implementation own AMI and stored in the AWS EBS for later launching similar instances with all the configuration already made. Using the AWS Free Tier offer determined the level of the launched instance; *t2.micro* incorporating 1 CPU with the clock speed of 3.3 GHz and 1 GB of memory. However, regardless of the low performance level, the according instance type it is perfectly suitable for the initial proof-of-concept implementation usage.

Running instance was set up with AWS Elastic IP (Static IP) for accessing the instance from the application level devices with immutable IP address. At the initial launching stage of the Amazon Linux a *.ppk* file was provided via AWS portal. According *.ppk* file was translated with PuTTYgen for *.pem* file to be used with SSH program (PuTTY) making the terminal connection with the instance. Via formed terminal connection, a Node.js environment was installed together with Node.js package managing software, *npm*. These environments are available for installation directly from Node.js and npm.js webpages with Linux commands illustrated in the following lines.

```
curl --silent --location https://rpm.nodesource.com/setup_4.x | sudo bash -
sudo yum -y install nodejs
sudo yum -y install gcc-c++ make
curl --silent --location "https://www.npmjs.org/install.sh" | sudo bash -
```

Another key functionality is the FTP server, hosting a file transmission from the robot into Amazon Linux instance no identified as TUT-AM-EC2 instance. A Linux FTP server called vsFTPd was used to fulfil this functionality. As for the Node.js environment, vsFTPd needs to be installed by the user. Installing the software occurs with another Linux command.

```
yum install vsftpd
```

vsFTPd needs additionally few altered parameters inside the vsFTPd configuration file located in */etc/vsftpd/vsftpd.conf* This configuration file can be altered with *vi* editor adding the following extra lines in the file.

```
# Additional configuration
pasv_enable=YES
pasv_min_port=port range min
pasv_max_port=port range max
pasv_address=Amazon Elastic IP address
local_root=root folder for the FTP server
```

As illustrated in the above lines, a local root folder is set up for vsFTPd server. This means that a user with the equivalent home directory and a password needs to be set in questioned Linux instance. Task was conducted with Linux commands of *adduser* and *passwd*. Transferring Node.js program files and testing the FTP server can take place with any FTP client program. FileZilla was selected for this purpose. FileZilla requires the address for the endpoint (either TUT-AM-EC2 elastic IP or DNS name), used protocol and *.pem* file for making the connection. Accessing the vsFTPd server, *.pem* file is not required, yet user root folder need to be specified keeping in mind that questioned folder is password protected. Additionally, when accessing the TUT-AM-EC2 instance or vsFTPd server, a used port needs to be configured in FTP client. With SSH port number 22 and with vsFTPd port number 21.

AWS RDS MySQL database was set up for storing the process data in structured form separate from the S3 bucket in which the data is stored as its native *.txt* format available for accessing with web browser by anyone granted with the permission. At the launching stage of the AWS RDS MySQL service a backup functionality was configured out of the usage. Reason for this was to save AWS EBS space for future use, such as additional stored AMI's. As it comes clear in the Chapter 4.4 a *.txt* file of the process data is always left intact in AWS S3 enabling the reset of the database in case of the data loss. Size of the database was set up for 10 GB and *t2.micro* instance (Free Tier offer) was selected as the platform. Portrayed in the Figure 14 and Figure 20, using the AWS RDS MySQL takes place via already running EC2 instance. Thus, database was bound to existing TUT-AM-EC2 instance and existing VPC. Further MySQL Workbench was installed and configured for accessing the database from user workstation. Function takes place by making the configurations in Workbench of TUT-AM-EC2 instance DNS name, password in the database and MySQL port number of 3306.

4.3 Real time process monitoring

In the Chapter 3.2 where application layer was introduced there was a mention of a DualCoat powder feeder. The installation of the powder feeder to become one part of the possible AM technologies in the cell, where delayed and was left for the future add-on. For this reason, the implementation part only covers CMT and COAXwire devices. Both of these devices hold their own monitoring software for true Real-time monitoring. Yet these are independently great set of tools, end user requires a convenient way of monitoring the ongoing process at the supervisory level. Supervisory level in here is meant as monitoring the whole production cell or the particular process in one view. Together with the AWS framework and Wapice IoT-Ticket technology, these objectives can be gained. Following two chapters takes closer look for the infrastructure and the architecture of the Real-time monitoring. Portraying starts with Chapter 4.3.1, the description of how process variables are gathered. Following Chapter 4.3.2, details visualization of these variables via IoT-Ticket Dashboard.

4.3.1 Gathering of process variables

Before visualizing the process, the adequate variables must be collected from the devices. Robot controls all the processes in the cell. Thus, gathering of the variables takes place via robot REST interface accessing directly for according variables in robot IO system's EthernetIP adapters. Additionally, the overall Supervisory status of the cell is monitored with robot's own RAPID program variables for controlling the questioned process. Regardless of the monitored process or monitoring the production at the Supervisory level, functionalities and the order of the operations are identical. Steps which are taken from the initial launch of the monitoring, for reaching the moment when the data is send to IoT-Ticket data server can be portrayed in the Figure 21. These steps are de-

tailed in following paragraphs concerning the monitoring of the production at Supervisory level, included with few words over the basis of the process monitoring.

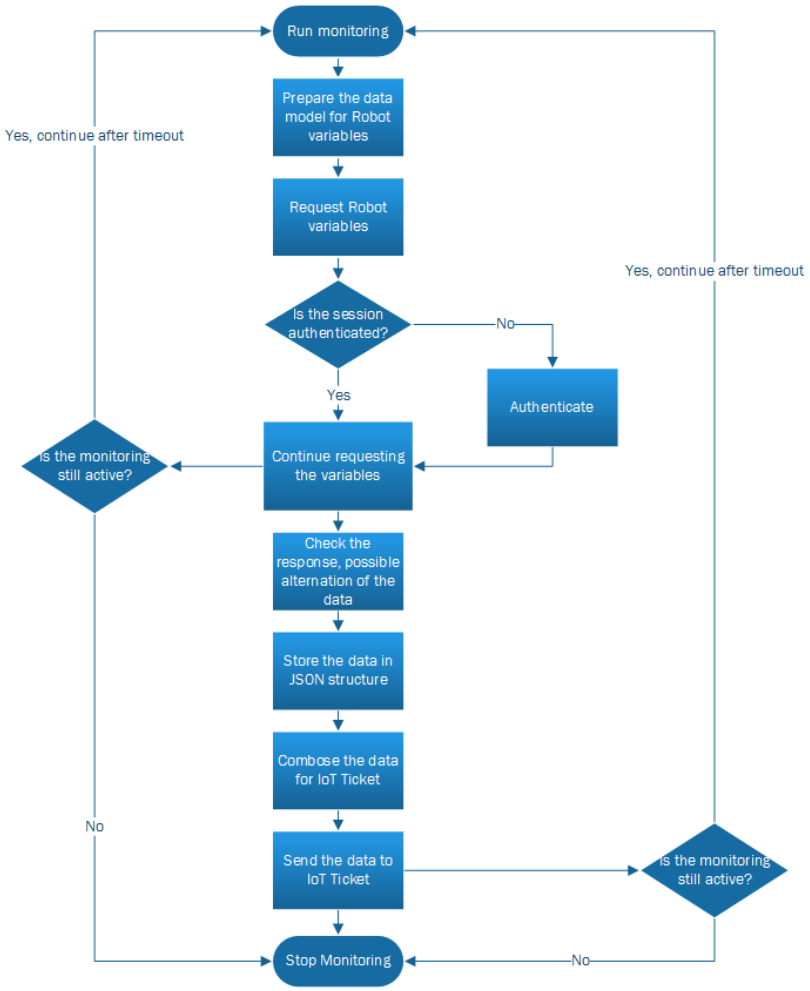


Figure 21. *Flowchart for Real-rime process monitoring*

Steps, portrayed in the Figure 21, are higher-level notifications of the software components used to perform the functionalities. Before continuing to description level of these operations, a following figure can be noted. Figure 22 portrays the program architecture deployed in TUT-AM-EC2 instance for collecting the variables and transposing them into IoT-Ticket platform.

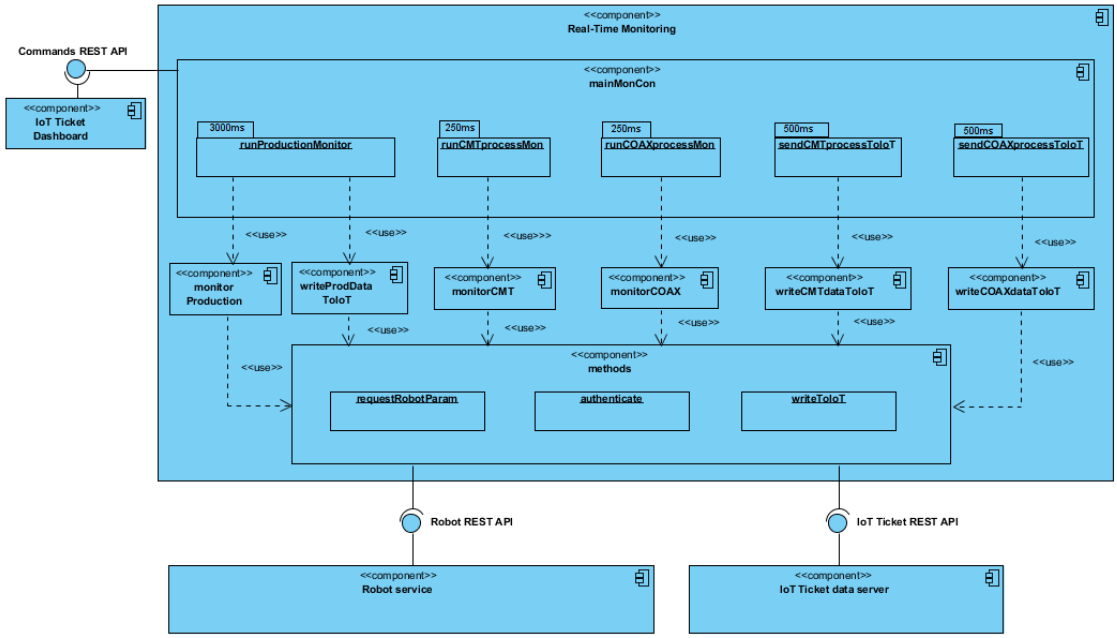


Figure 22. *Program architecture for Real-time process monitoring*

Portrayed in the Figure 22 the entirety of the process monitoring can be realized as one independent component (**Real-time Monitoring**), composed from several individual software components acting together by program calling and using dependencies from outside services with REST interface. During the implementation, a future prospect of shifting the programmed backend environment to be executed within resources of an alternative service provider was kept in high value. For this reason, the individual operations where heavily modulated. All the modulated components inside the main component, are deployed in one folder inside TUT-AM-EC2 instance. The main module is executed with Node.js *forever npm-module* for evading rebooting of the instance. In the following, functions of specific components are detailed.

mainMonCon.js is the main component launched at the start of the Real-time monitoring. It holds all the sub-functions for calling appropriate operations for the monitoring. After the start, according component instantly calls *runProductionMonitor* function and is left waiting for commands via IoT-Ticket Dashboard REST API (*Commands REST API*) which is used when launching CMT monitoring or COAX monitoring functions. IoT-Ticket Dashboards are configured to hold button elements for initiating a monitoring for according process (see Chapter 4.3.2 for details). By pressing these elements user sends a HTTP POST request with HTTP basic authentication to TUT-AM-EC2 instance for activating the monitoring. With similar elements, a user can stop the monitoring. Questioned requests are described in Table 9. Furthermore, this component has logging capabilities with time and date record for later debugging the activities. Similar debugging is also available with all the other components portrayed in Figure 22. Logging was performed with *log4js* module. log4js was selected from the reason that it holds interface for logging inside Loggly service [171]. Loggly is a cloud based logging

management service where logs are able to be accessed and studied online via Loggly web pages [172].

Table 9. *HTTP request for controlling the monitoring*

Function	**URI** **base-URI:** http://TUT-AM-EC2-Elastic IP:PORT/	**Method**
Start CMT process monitoring	cmt/startmonitor	POST
Stop CMT process monitoring	cmt/stopmonitor	POST
Start COAX process monitoring	coax/startmonitor	POST
Stop COAX process monitoring	coax/stopmonitor	POST

runProductionMonitor function is launched at the start of the *mainMonCon.js*. Function is left running with timeout interval of 3 seconds, which becomes the update interval for these gathered variables. Function controls the additional components for monitoring of the production inside entire application level (Supervisory level).

There are couple of reasons for dividing the functions and components in the form as described in Figure 22 is that different components are handled with distinct timeout intervals. These timeouts are represented in the questioned figure with millisecond records above the function. Timeouts can more easily be handled at the upper level of the architecture than encapsulating within individual components. Production can be monitored with less frequency than the processes itself. Use of the production monitoring is to give a user basic information over the running application, concerned information is detailed in Chapter 4.3.2. Additionally, the lower level components can easily be maintained and modified.

runCMTprocessMon and **runCOAXprocessMon** -functions are launched when the user requests for according process monitoring via IoT-Ticket Dashboard elements. These functions are driven with 250 millisecond timeout interval and them use *monitorCMT* and *monitorCOAX* components. Compared to foregoing functions **sendCMTprocessToIoT** and **sendCOAXprocessToIoT** are called with 500 millisecond timeout interval. These two functions control the data sending for IoT-Ticket Data Server which holds maximum of 500 millisecond update interval. Thus, variables itself are updated within backend service with higher frequency for gaining more accurate data ready for to IoT-Ticket provision.

monitorProduction component is used by *runProductionMonitor* function for First; preparing the data model for accessing robot variables via Robot REST API and Second; storing requested variable data with JSON format within the component. Component uses the pre-prepared JSON data structure where the URI's for desired robot variables are stored (see Table 10 for details). These URI's are transposed for sub-component of **methods** where *requestRobotParam* function sends the HTTP GET request with

specified URI, by *npm request* module for Robot REST API. In the case where the session is not established, robot returns with HTTP error code 401, not authenticated. Component responses with function *methods.authenticate* and returned digest authentication cookie is stored in the component's *cookie-jar* for later accessing the API. When the authentication is verified, responses from requesting the variable states are stored in JSON structure. In the case where additional formation of the returned variable is needed it is handled before storing. This enables immediate later transpose of the variable data for IoT-Ticket Data Server.

Table 10. *Robot variable URI's for monitoring the production*

base-URI: http://robot-ip-address/rw/rapid/symbol/data/RAPID/T_GUI/CellData			
Name	**resource-URI**	**Datatype (IoT-Ticket)**	**Unit (IoT-Ticket)**
timeETA	/sTimeETA?json=1	String	Text
currUser	/sCurrUser?json=1	String	Text
process	/sProcess?json=1	String	Text
procedure	/sProcedure?json=1	String	Text
details	/sDetails?json=1	String	Text
platformID	/sPlatformID?json=1	String	Text
fixture	/sFixture?json=1	String	Text
timeStarted	/sTimeStarted?json=1	String	Text
prodAvailable	/bProdAvailable?json=1	Boolean	bool

Current implementation is though a narrow introduction for the subject and for this reason, each device inside IoT-Ticket data structure (under the enterprise level) in added manually via specifically formed JavaScript code. From the response of the new device formation, an identification number is grasp and stored in JSON file, *devices.json*. Path of the each Datanode is established by the user and them are stored by IoT-Ticket server each time current Datanode-path combination is initially written to data server (see Chapter 3.4.1 for details). These Datanode-path combinations remain evermore the same. Thus, them are saved in another JSON file, *datanodes.json*. Datanodes file holds the name of the node, path for the node, unit of the node and datatype of the according node. Glimpse of the file is portrayed in below lines of JSON structure.

```
"prodTimeETA":
      { "datanode": "prodTimeETA",
        "path": "Production",
        "unit": "text",
        "dataType": "string"
      },
"prodCurrUser":
      { "datanode": "prodCurrUser",
        "path": "Production",
        "unit": "text",
        "dataType": "string"
      }
```

Both the *devices.json* and *datanodes.json* are accessed by the **writeProdDataToIoT** component. Initially current component assembles a JSON structure, which is a combination of variable data stored by *monitorProduction* component, and Datanode information from *datanodes.json*. Data and information are stored in secondary JSON structure in a way that all the retained data can be send within one single message. A glimpse of the mentioned structure is illustrated below this paragraph. Together with device ID and the formed data structure mentioned component uses *methods.writeToIoT* function for sending the data to IoT-Ticket data server. **methods.writeToIoT** holds authentication information, method (POST), headers and the base URL for accessing the data server. Final URL is formed from the base URL and from JSON structure (*path*) transposed from *writeProdDataToIoT* with component call.

```
var dataToWrite = [{
      "name":     datanodesJSON.prodTimeETA.datanode,
      "path":     datanodesJSON.prodTimeETA.path,
      "v":        productionData.timeETA,
      "unit":     datanodesJSON.prodTimeETA.unit,
      "dataType": datanodesJSON.prodTimeETA.dataType
      },
      {
      "name":     datanodesJSON.prodCurrUser.datanode,
      "path":     datanodesJSON.prodCurrUser.path,
      "v":        productionData.currUser,
      "unit":     datanodesJSON.prodCurrUser.unit,
      "dataType": datanodesJSON.prodCurrUser.dataType
}
.
.
.]
```

When monitoring one of the process (CMT or COAXwire) data model and the data flow takes similar steps as described above. Difference comes only from the initial operation where user need to activate the monitoring via IoT-Ticket Dashboard, for the timeout interval of the data update and for the gathered variable data from the robot. For these reasons, detailed operations for process monitoring are not described. Yet, viewer can find the names and the specific Robot URI's for gathered variable data in Appendix B.

4.3.2 Process variables visualization

After the data is gathered from the robot and sent for IoT-Ticket Data Serv, a creation of the Dashboards can take place. Customer can hold as many Dashboards as they keep necessary and movement between these Dashboards are handled with IoT-Ticket user interface Control Pane or with hyperlinks inside each Dashboard. Creation and modification of the Dashboards are conducted with online Interface Designer together with Dataflow Editor for making the evaluation and modification for the data before representing it for the viewer. According chapter takes closer look for Dashboards used in the final solution briefly rising some detailed information from the phase of building the Dashboards of the project.

Gathered real-time visualization variable data is structured inside IoT-Ticket Data Server under the enterprise level and furthermore under each process title. Described categorization, further portrayed in the Figure 23, holds an advantage when using datanode (datatag) selector tool in Interface Designer or in Dataflow Editor. Datanode selector tool can represent the nodes as a list containing all the nodes, yet another possibility is to choose the right datanode from the tree structure. While the detailed hierarchy is adopted and when building the Dashboards, all the numerous datanodes are not needed to go through for finding the right one from the certain process.

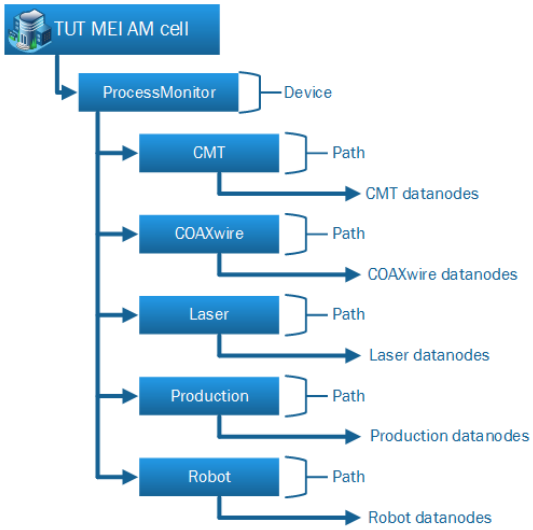


Figure 23. *IoT-Ticket Datanode architecture for Real-Time process monitoring*

After logging into IoT-Ticket webpage, the user is provided a hyperlink for the production Dashboard. This Dashboard represents the Supervisory level status of the application environment (Figure 24). At the top of the Dashboard user is eligible for starting and stopping of the monitoring for this particular view. Operation only activates or deactivates the updating of the page; the questioned variables are still gathered via robot REST API and stored inside IoT-Ticket Data Server. Described action is created for lower data rate internet connections which might cause blocking of the end device when

updating the view each 3 seconds, set for now as update interval (controllable via Dataflow Editor, at the moment same as the variable gathering timeout interval). Below the update control, a button element is presented for selecting the desired process for detailed monitoring (CMT / COAXwire). By pressing the button, a prompt screen is visualized and user can make the selection. At lower part of the initial view, the actual variables are presented with their current values. Under the progress bar, a button element for accessing History Data is available. Another button is given for triggering a report from the latest finished process run. At the pages' footnote, IoT-Ticket Control Pane is available with additional Dashboard page navigation elements.

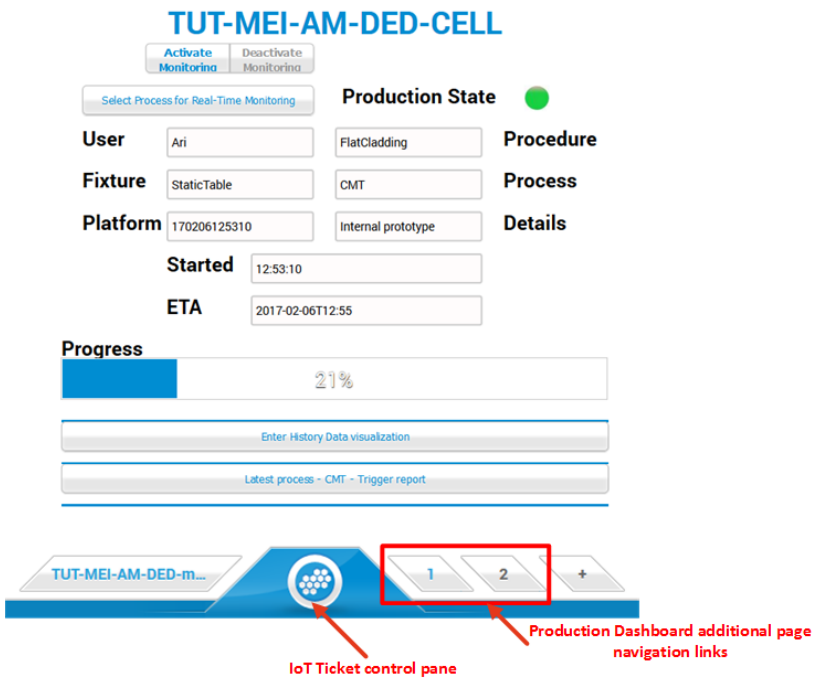


Figure 24. *Production monitoring Dashboard*

Selecting the desired process, user is directed over to the main page of the according Dashboard. These main Dashboard pages varies over the selected process. CMT process Dashboard main page holds only meters as indicators at the middle section. Whereas COAXwire monitoring Dashboard main page holds 4 meter elements and two charts. One for monitoring the robot TCP (Tool Center Point) velocity and one for wire feeding velocity. Figure 25 portrays the main page of the COAXwire monitoring.

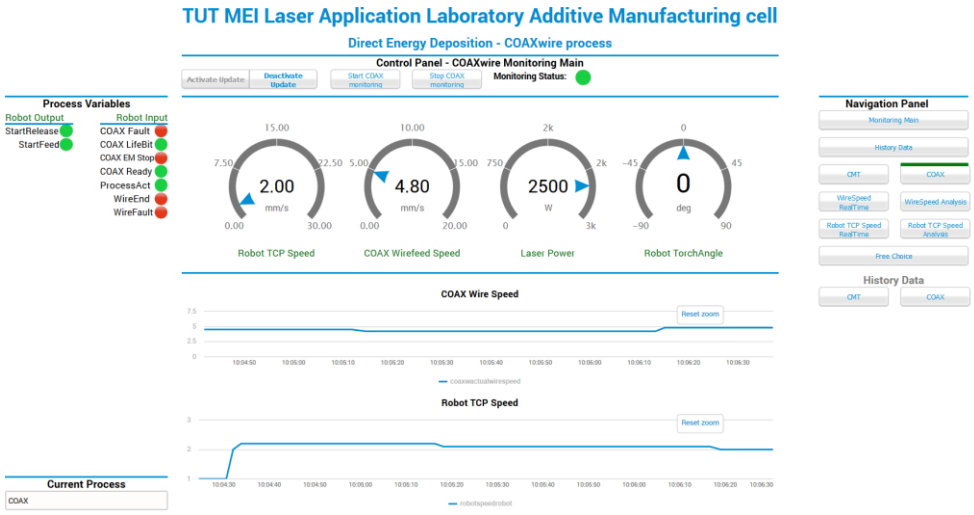


Figure 25. *COAXwire monitoring main Dashboard page*

At the top most position of the Dashboard, there is a control panel. Via buttons of the according panel, the user can activate or deactivate the page updating, similar as for production monitoring page for low rate internet connections. Alongside these buttons user can start or stop the variable gathering itself. By pressing the *Start COAX monitoring* –button IoT-Ticket sends HTTP POST message for TUT-AM-EC2 instance REST API, requesting the start of the monitoring. TUT-AM-EC2 replies accordingly and *Monitoring status* light is turned green or remains red, in the case of failure. Feature is created with IoT-Ticket button element widgets "*drag'n dropped*" inside the Dashboard working canvas and connecting these elements inside Dataflow Editor with IoT-Ticket REST API widgets. These widgets are configured to hold the URI's and port of the TUT-AM-EC2 REST API with additional HTTP basic authentication username and password. For providing the viewer a glimpse of the operations within Dataflow Editor described configuration is portrayed in the Figure 26.

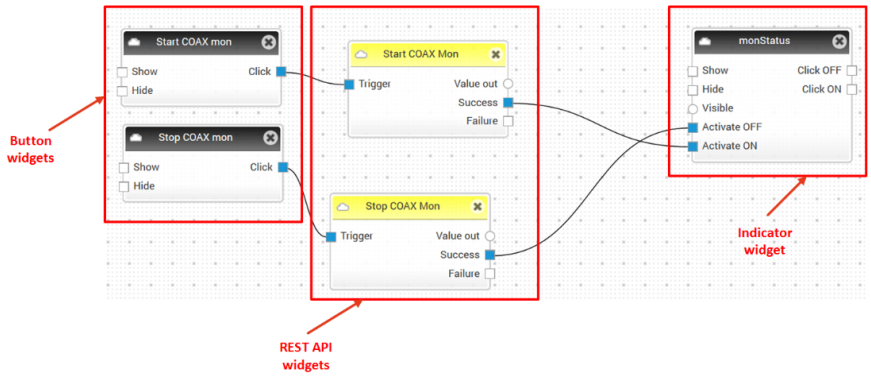


Figure 26. *Dataflow Editor setup for starting and stopping of the COAXwire variable gathering*

At the middle section of the COAXwire main Dashboard (Figure 25), meters for monitoring robot TCP velocity, wire feeding velocity, laser power and the angle of the

COAXwire tool, are present. Below the meters, a set of two charts are located for brief history glimpse of the according variables. This helps the user when monitoring the corners and the edges of the artifact manufactured with COAXwire process. On the left hand side of the page, a group of selected Boolean variables is gathered under the categorization of their existence from the aspect of the robot. Another feature providing the viewer a top down look over the manufacturing of the artifact. Finally, at the right hand side a *Navigation Panel* is formed for making hyperlink jumps between desired Dashboards. Green line above the button indicates the current page. Initial button named *Monitoring Main* redirects the user for *Production monitoring* main page (Supervisory monitor). *History Data* button is used for switching inside visualization of the already finished processes. More of this feature in described in the Chapter 4.4.3. *CMT* and *COAX* buttons switch between the desired process monitoring main pages, respectively. In the middle, there is a set of four button elements. Use of these redirects the user for sequential pages below the main Dashboard of the current process. These pages holds graphical charts where user can make the selection of specific timeline for observing the history of the according variables. Selected data is traced from the real-time monitoring variables. Reason for the existence of the functionality is to eligible the user a glimpse into the history of the process while it is still running. Feature arises from the method where actual history data is collected by the robot and transposed into IoT-Ticket Data Server moments after the process is finished. *Free Choice* button element on the *Navigation panel* redirects the user for a history glimpse page where user can freely choose the variable for detailed observation. Possibility for Free Choice selection was a user experience feedback. One additional feature within the real-Time monitoring Dashboards are the short cut elements to instantly relocate the user inside *History Data* observation.

For the CMT process monitoring, main Dashboard appears much of similar aspects with COAXwire process. There are difference coming from distinct monitoring variables and features over the process itself. With CMT process there are no charts available set forth for the user at the main monitoring Dashboard. Values of the individual variables hold more benefit for the user when illustrated in transient mode with meters. Like in COAXwire process, CMT monitoring holds sequential Dashboards for taking a peek within the history of the according variables and one for Free Choice of the variables. Main Dashboard for CMT process is portrayed in the Figure 27.

Figure 27. *CMT monitoring main Dashboard page*

Observing the history of the real-time monitoring variables take place equivalently regardless of the monitored process. By selecting the desired variable and method with the *Navigation Panel* buttons according Dashboard for history glimpse is prompted. The history observation is divided in two methods, for longer period of constantly updating real-time data and search of the real-time data from the past. For the CMT process, the real-time history data holds current and voltage variables. For COAXwire monitoring similar variables are robot TCP velocity and wire feed velocity. Additional variable selections can be maintained and altered by adding a new Dashboard page and configure the page according a new set of variables. Action, which is performed by the administrator of the solution while users can only access the interface for employing the graphics. Figure 28 represent the CMT process voltage visualization with constantly updating chart values. User can access this Dashboard by *Voltage Real-Time* –button. Chart at the top level of the Dashboard is updating the values in real-time and below chart is showing the variations of the minimum and maximum values with additional calculation of the average value. From the cited figure, it can be observed that process voltage has lowered during the inspected timeline. This information corresponds to lower chart indicating that minimum and maximum values stabilizing. Knowing the questioned process run it can be noted that wire-feeding velocity was deliberately dropped after first layer of the AM process. This affects the output voltage of the CMT device, now able to be observed from the charts.

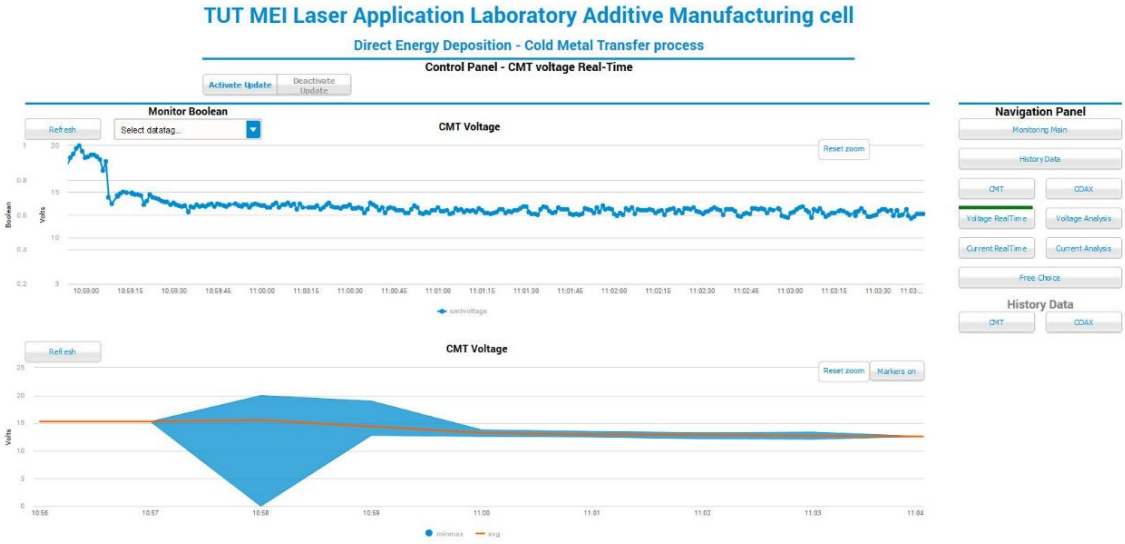


Figure 28. *CMT process Real-time voltage variable history glimpse*

Additionally, user can make a selection of the certain real-time variable analysis. In the case of voltage variable user can access this page by pressing *Voltage Analysis* – button in the *Navigation Panel.* Despite the name of the functionality, analysis ensemble of the data and variables are left for the future work conducted after the finalization of the thesis. Analysis in this relation is understood to be handled by the users in pure visual manner. Each variable analysis Dashboard is formed of upper and lower segments. At the upper segment user is provided with two date and time selectors, *From* and *To*. After choosing the desired start and stop moments for the visualization, a chart is traced accordingly. Lower segment of the Dashboard works in similar manner as the upper one. Lower part is used for visualization of minimum, maximum and average values from the selected timeline. Figure 29 portrays the functionalities described for the analysis Dashboards. Figure represent one short AM process. In the middle section, voltage was at stable state, indicating stable process. When closing the end of the timeline voltage was drifted out of balance. Process was shortly after halted for the cause of wire feeding error.

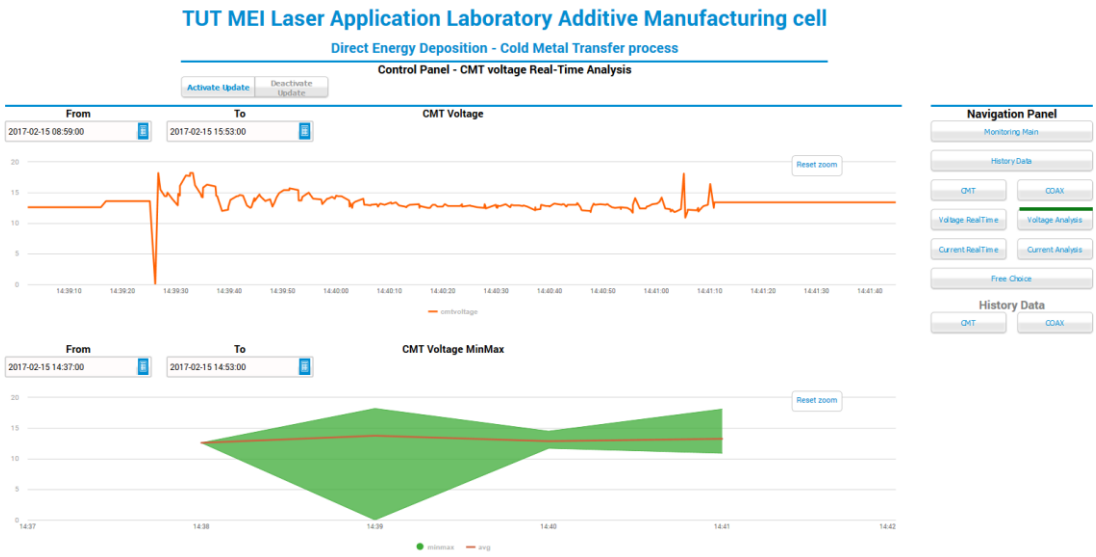


Figure 29. *CMT process Voltage variable analysis Dashboard*

Ascend from the fact that current and voltage in case of CMT and TCP velocity and wire feed velocity in case of COAXwire are the two main essential variables represented with real-Time monitoring history glimpse, a one-click relocation for these variables inspection were created. Additionally there raised a compulsion from the user experience feedback for freely select the variables for detailed history glimpse observation. To fulfill the need, one Dashboard page were designed and inserted with both CMT and COAXwire Dashboards groups. Within both of these Dashboards, a user can freely select out of two datanodes and one Boolean state to be visualized. Visualization is possible with both traced values and minimum, maximum and average values. Described Dashboard is portrayed in the Figure 30 as it is accessed in CMT process. Charts in the Figure 30 visualize wire feeding velocity values. Process in question has lowered the velocity of the wire feeding. This is due to gaining better contact with initial layers for the base material (build platform). After first layers, the process is stabilized with levelled wire feeding velocity.

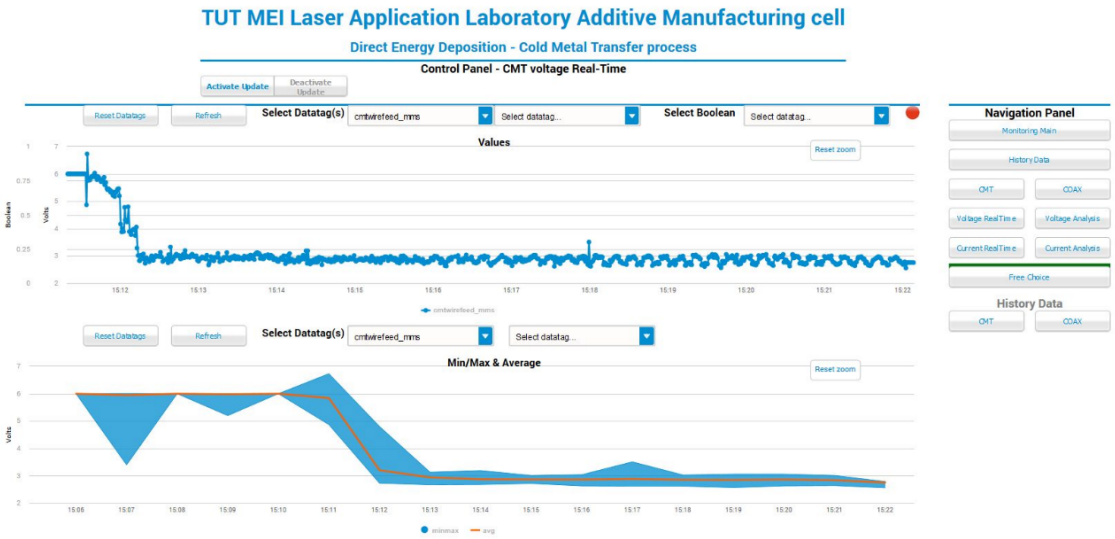


Figure 30. *CMT process Free Choice variable visualization*

4.4 Process data history

During the active manufacturing process robot is the controller for the application and thus storing process data in the background task. Data is stored with higher frequency than real-time monitoring takes place and so worth basic *.txt* file inside robot controller is the most convenient location for the retained data. After the processis finished collected data is transposed into TUT-AM-EC2 FTP server. Inside TUT-AM-EC2, the designed backend program acknowledges the arrival of new file and the manipulation of the data for storing inside AWS services and sending to IoT-Ticket Data Server can commence. Finally gathered process data is available for the users to be observed from IoT-Ticket custom created History Data Dashboards. In addition, a report is created from the current manufacturing process. In following Chapters of 4.4.1, 4.4.2 and 4.4.3 these operations are detailed, respectively.

4.4.1 Process data integration and transfer

Starting of a new artifact production with a certain process, that is starting manufacturing with application environment, robot goes through the setup chosen by the user (robot user interface, see Chapter 3.2 for details). With these settings, desired process is started. Another possibility for the user is to create robot program, either with offline or online programming (see Chapter 3.2 for details). Regardless of the initial input of how the process is started, robot recognizes a change in a specific Boolean variable set active at the start sequence. In consecutive phase controller checks the type (CMT or COAX-wire) of the process and forms a new *.txt* file accordingly. Controller continues to written variable data in line-by-line basis.

Mentioned *.txt* file is based on three distinct segments. At the initial segment (line) a key identification notes for the questioned process run are noted. On the second segment each line is led by the timestamp, followed by the gathered variables data. Final segment is the last line in the file. According line contains time when process was finalized and a notification of seized process. Viewer can review the tables in Appendix C, for more detailed information over the content of the segments. Appendix A furthermore describes the meaning and the usage of the gathered variables in the process. An example of actual process data file in case of CMT process can be viewed from Appendix D.

When booting the robot controller it connects to TUT-AM-EC2 instance vsFTPd server with FTP client and mounts the directory inside the controller. After the process is finished the mentioned Boolean variable as for starting the data gathering is set inactive and together with inactivation of the process (CMT or COAXwire), the final segment is written for the file. Middle segment of the composed file contains now process data, written with the frequency set by the user. Robot moves the file for mounted vsFTPd server directory. This action further automatically transposes the file into AWS TUT-AM-EC2 instance FTP server directory. Robot re-enters for waiting a new process and the software components inside TUT-AM-EC2 instance continues for the file manipulation.

4.4.2 Process data manipulation

Process data files are transposed in specific folder inside TUT-AM-EC2 instance. Questioned folder holds own username and usage rights and it serves only for the purpose of the data files. Folder is monitored with Node.js backend program. New file is shifted over to Line-by-Line reader and handled according to line count. Steps from the new file arrival until the file is entirely processed are detail in the Figure 31.

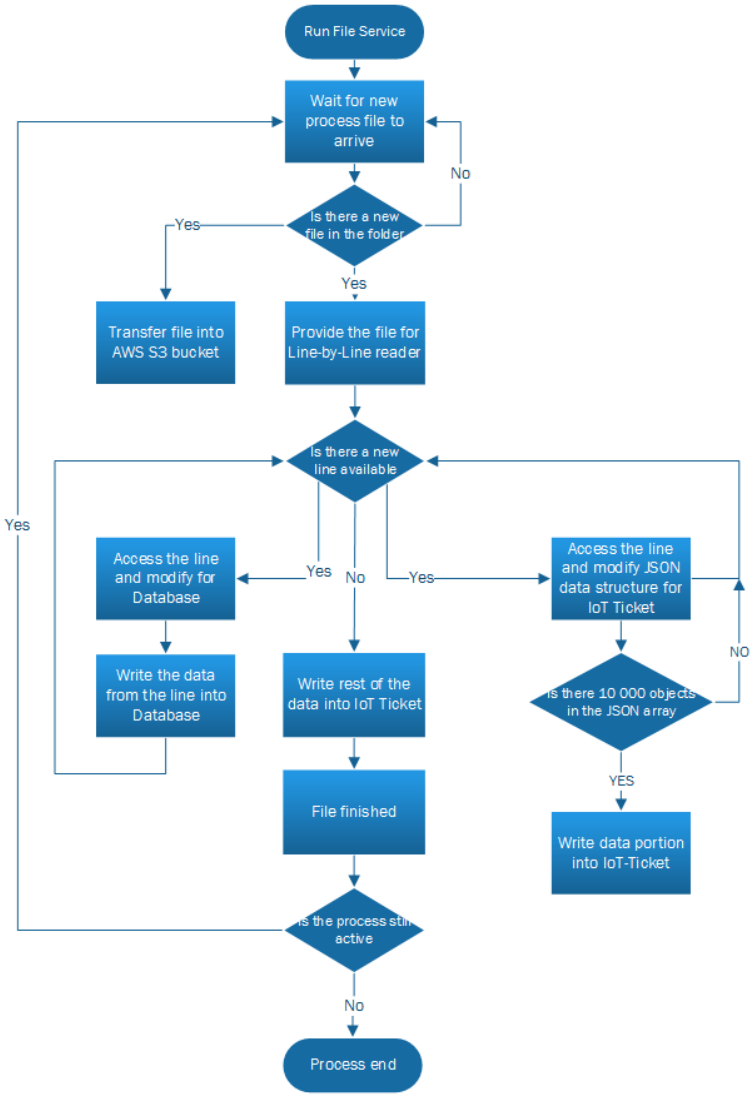


Figure 31. *Flowchart for History Data storing processing*

Viewer can notice from the above figure that after the line is read it is processed with two independent threads. One for writing the records inside AWS RDS MySQL database and one for forming the data model over the records, to be send into IoT-Ticket Data Server. For more detailed knowledge, viewer can look for Figure 32 (below) where the program architecture for file processing is portrayed with Unified Modeling Language (UML) chart. As it can be observed, architecture is more elementary when compared with real-Time monitoring.

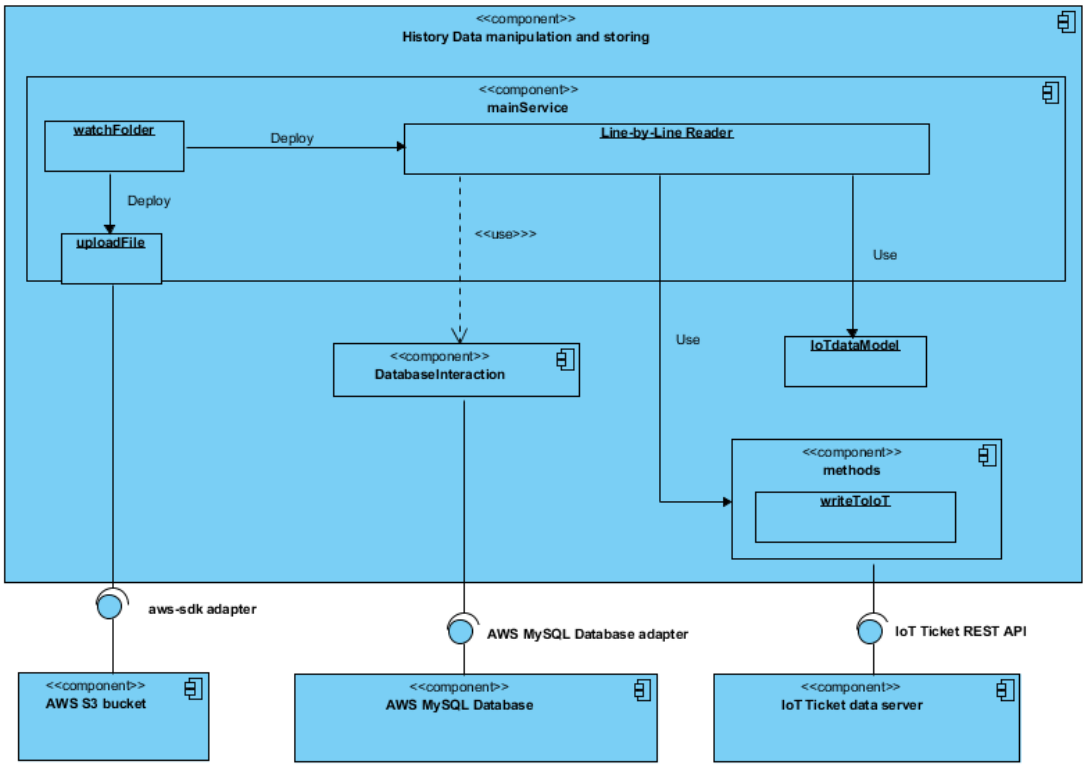


Figure 32. *Program architecture for History Data processing*

Portrayed in the Figure 32 the entire file processing can be comprehended as one component (**History Data manipulation and storing**). This component is formed from multitude of individual components acting together for outcome. All the software components are deployed under *FileService* folder within TUT-AM-EC2 instance and they hold *log4js* logging feature. At the initial phase, **mainService.js** is launched with Node.js *forever npm-module*. Inside *mainService.js* a **watchFolder** function is deployed for monitoring the changes in the FTP server dedicated folder. Function is implemented with *npm* module called *watch*. *Watch* module can notice a multitude of changes, yet in here, a new file arrival was accessed. After a file arrival, *watch* module provides the file name for the function. According file is moved inside AWS S3 bucket with Node.js *aws-sdk* module. At the same time file name is provided for **Line-by-Line** reader function, developed around Node.js *npm linebyline* module.

Line-by-Line reader commences accessing the file. According to the line count, a specific manipulation is carried out. At the initial line, there is a basic information for the finished process. One important record inside this line is the knowledge of the according process (CMT or COAXwire). Database interactions and IoT-Ticket interactions are handled accordingly. From the line, a start time is taken out and shifted for millisecond record. Reason for such action comes from the IoT-Ticket Data Server where timestamps are handled with unix-time (epoch time) at millisecond accuracy. Together with rest of the data from the initial line, a data model is structured for later sending the

basic process information to IoT-Ticket Data Server, under *ProcessData* device. One additional data model is formulated to hold the start and the stop times of the process with millisecond form. This data model is transposed into IoT-Ticket after the file handling is ended. Reason for two distinct data models is to update the mentioned timestamp values within *Production* -path in *ProcessMonitor* device. Two different device interactions requires two different REST requests. Recognition of the first line also immediately updates another Boolean variable inside *ProcessData* device. Updated variable is chosen according the process the file concerns. These timestamp values and Boolean variables are later accessed when triggering a report creation. More detailed information can be found from Chapter 4.5. Simultaneously the first line of the file is provided for **DatabaseIntercation** component. Database is constructed with one-to-many architecture as illustrated in the EER (Enhanced entity-relationship) model in Figure 33. Basic information from the process is stored inside *Production* table, where *buildPlatform* is the primary key. Data from the lines holding the process variables are stored within according *buildPlatform* table. A new *buildPlatform* table is formed each time new process file is accessed. Build platform is the identification of certain process run. This identification is formed by the robot when starting a new process. Format comes from the current date and timestamp. This enables each process to be identified with distinctive ID. Handling the initial line of the file, *DatabaseIntercation* component formulates the SQL query and writes the data inside AWS MySQL database *Production* table.

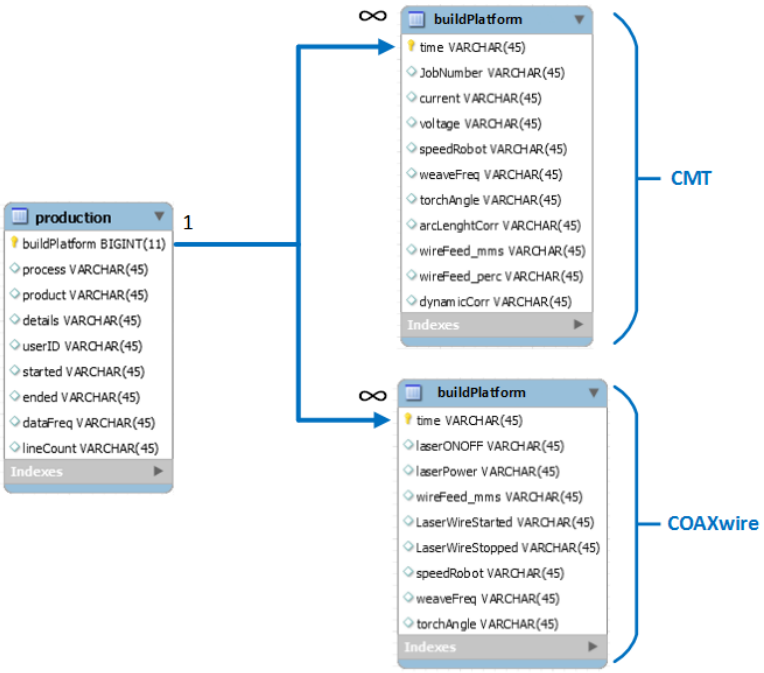


Figure 33. *Process data - database structure*

Sequential comes the second line of the file. This line is the first holding the variables from the process. Line is provided for both **DatabaseInteraction** component and

IoTdataModel function. *DatabaseInteraction* component verifies the line count; forms SQL query and stores the data inside according *buildPlatform* table. Connection with IoT-Ticket Data Server is slower than internal connection with AWS RDS MySQL. For this reason, a data model is formed to hold the data from the process records in JSON array structure. This structure is monitored and each time it reaches 10 000 JSON array objects the data portion is sent for IoT-Ticket Data Server, within the *body* of the HTTP message. After the file handling is finished, remaining data in JSON array is sent for IoT-Ticket Data Server. *IoTdataModel* function handles the formation of the JSON structure. With the function call, a process start time in unix-time form is provided. Function excavates the timestamp (milliseconds from the start of the process) from the provided process line and adds the value with provided start timestamp. A timestamp is now calculated for all the records within the line. Function forms a temporary JSON structure for each record. These structures are then pushed inside another JSON array for holding the records. Reason for such double array action arises from the feature where IoT-Ticket datanodes are handled with identical keywords; each datanode in that case need to be handled individually.

Each line holding the process variables are processed similar way, database is updated and JSON array is pushed with records from the line, while monitoring the size of JSON array. As the file is handle, *Line-by-Line* reader watches every line for key word 'end'. From this record, a final line is realized and stop time for the process is found after the key word. Database *Production* table is now updated with according time. Equivalently the data model for the basic information (for Iot-Ticket), is now updated with end time. Simultaneously a second data model for holding the start and stop times in millisecond form and data model for holding variable information for triggering a report creation is updated (for IoT-Ticket). All data models are now completed and records are send for IoT-Ticket Data Server via *methods* component. Administrator of the solution has beforehand prepared a device inside IoT-Ticket Data Server for holding the process data. Returned device ID is stored inside *devices.json* file. Actual data is stored with Datanode-path combination, similar as for the real-Time monitoring data. These Datanode-path combinations are stored inside *datanodes.json* file.

4.4.3 Process data visualization

Designing and building of the IoT-Ticket Dashboards for process History Data takes similar steps as for real-time process monitoring described earlier. Thus, following chapter takes into consideration only the details exclusive for History Data Dashboards and visualization.

For the reason of the the IoT-Ticket Data Server structure, few design rules must be applied when storing the data. A new device, *ProcesssData* was formed to hold the data for finished processes. According device is used for making the separation over the

ProcessMonitor device serving as real-time monitoring variables. The understandable difference welling from the structure of sorting the two distinct data and method purposes. Inside the *ProcessData* device data is structured under the two different processes, CMT and COAXwire, respectively acting as IoT-Ticket *paths*. Basic information of the finished processes is further stored under each process *BuildplatfromID* path. Every new finished process is given a *BuildplatfromID*, formed similarly as database forms *buildplatform* primary key and table. These ID's are turned as datanodes holding the according information. Each questioned process path holds the datanodes for all the gathered data. Same datanode holds the data for each of the processes. It would have been conveniently to produce a new datanode for each gathered data record after the each finished process. Furthermore, the user could have visualized these datanodes accordingly. Unfortunately, IoT-Ticket does not hold a *datatag* selector widget with a tree structure option, accessible in finished Dashboard. Without a tree structure visualization, user would be forced to go through all the datanodes when searching the right process in question. Possible update is been revised, yet the option is not present at the current moment. Final History Data storing architecture is illustrated in the Figure 34.

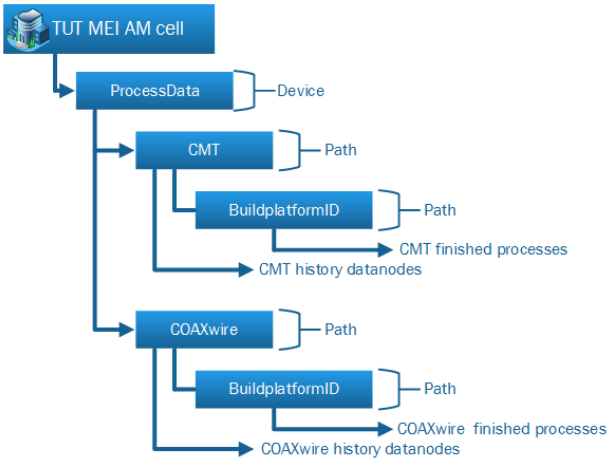


Figure 34. *IoT-Ticket datanode architecture for process History Data storing*

From the production Supervisory monitoring page (Figure 24), a button element is found for shifting into process History Data visualization. After reaching the button, initial page for the user is the main History Data visualization Dashboard. This page holds a summary of the finished processes and button elements for accessing either CMT or COAXwire for more detailed visualization. Additionally, a dedicated button can be used for returning the Supervisory monitoring page. Regardless of the accessed process, the operations within the History Data visualization are similar. Only change takes place with the gathered data. For this reason following considers CMT process with more details.

Dashboards for finished processes History Data are composed of four different sections and two distinct Dashboards are provided for the user. First of these Dashboards is illus-

trated in the Figure 35. Figure inside black rectangle is the one visualized for the user. Additionally some parts are highlighted for better viewing. On the upper section, there is a table for portraying the critical information over the finished processes. On the left column of the table, ID's of the process correspond on the right column holding the details. These details are represented in the Appendix C (first and last line of the process file).. Middle segment is used for making the selection over the variable data. User can access two buttons, one for resetting the datatags selection(s) and another for refreshing the selected datatags after the made selection(s). Selections for requested data are handled with dropdown menus. User can select maximum of two datatags with one Boolean variable. Adding more data with one chart would make the chart unreadable. *From* and *To* date selectors are used for viewing specific time in the process. User first searches the desired process from the above finished process table and inserts the desired time values with date selectors. Lowest part of the Dashboard holds the line chart for representing the values of the data. Finally, on the right hand side a *Navigation Panel* is located. Green line above the button gives the information over the current selected Dashboard. Additionally, green color was selected to act as headline color within History Data Dashboards. Manner, which gives user more briefly the information of the located functionality.

For the actual process, Figure 35 portrays a situation from the application tests where real-time monitoring revealed that some error occurred with CMT current value during the initial layers of the artifact. After the process was finished and data inserted into IoT-Ticket CMT current was studied. A drop of the current was indeed located and further examination was conducted view IoT-Ticket 'zoom' feature. CMT current was observed together with wire feeding velocity. Combination of these two unveiled that wire feeding was dropped near zero and moments later current was dropped to zero. Occurrence corresponds with knowledge where current must be lowered if no wire is fed. When counting the samples and knowing that data storing frequency was 100 Hz (10 ms) wire feed was at zero level for total of 260 ms. Occurred phenomenon did not reveal at any way in the final artifact. Yet, example indicates the usage of the real-time monitoring together with History Data observation.

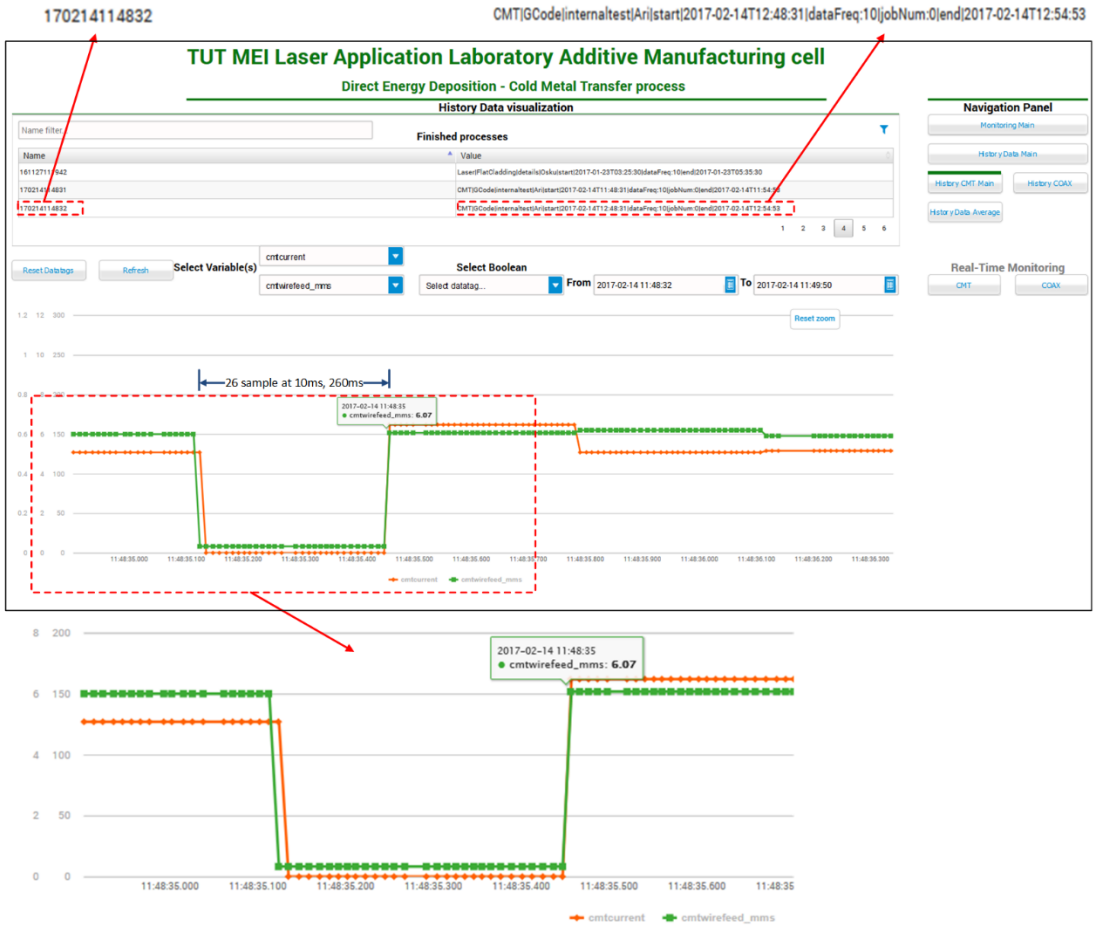


Figure 35. *History data plain values visualization for CMT process*

User has additional Dashboard for viewing minimum, maximum and average values of the datatags. Tracing of these records takes place with similar user interface as the case was with plain values visualization page. According Dashboard is portrayed in the Figure 36. Conceivable variables (datatags) are now reduced in one. With more variables in the same graph would make the chart unreadable. Viewer selects the time range with date selectors and datatag with variable menu. Refresh button is accessed for tracing the chart. Illustrated graphs shows minimum and maximum values as marked area and average values with plain line records. Questioned figure illustrates a situation where application is started and CMT current minimum and maximum values are fluctuating. This is due to the both process itself and user altering the wire feeding velocity for gaining more solid attachment for the artifact to the base material (build platform). During this short period of application test value is stabilized. Similar observation can be gained from the average value; line representation. Average value is lowering to the level where stable process might be continued. Unfortunately for other reasons this process was halted.

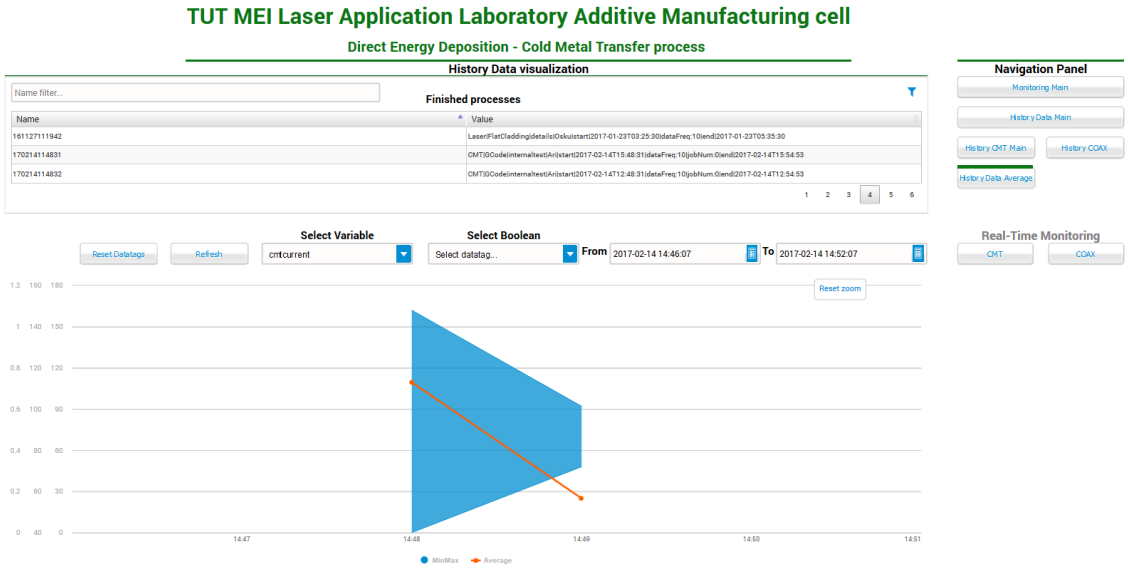


Figure 36. *History data average and minimum-maximum values for CMT*

4.5 Process report creation

This chapter describes the steps, which were taken for creating the reports for the application tests. Reports can be triggered on various occasions of the process. However, most conveniently, a report should be created and triggered for the finalized process. Although a start of a new process is additionally valuable information for the main users. When a shared electronic mailbox is harnessed, everyone can be provided with the information over the activity of the environment. Thus, a report was designed for start of a new process and for finalized process. Following describes in total three types of reports designed. Viewer can look for more visualized information from the Appendix E that holds the examples of all the reports.

Designed reports were elected to hold the basic coverage of the according process. These layouts are going to be updated and maintained as more information over the parameters and process itself can be gained during the future usage of the application environment. As for finalized process report, a coversheet starts the report after which comes the actual datasheets. Coversheet is mainly a static page holding company name and logo. Dynamic feature located on the coversheet include a timestamp for report creation. Both of the processes, CMT and COAXwire hold their specific report layouts. Dynamic information changes according each process. For the CMT application, page following the coversheet, holds an overall information of the process. Similar variables already represented at the main real-time monitoring Supervisory Dashboard. CMT report continues with representing the average current and voltage accessed during the manufacturing, including furthermore minimum and maximum values of the variables. Report is moreover used for maintenance purposes for the CMT device. Maximum value of the wire feeding rollers current is calculated and represented. In the case where

value crosses above the selected threshold report provides the information for the user to consider inspecting the wire-feeding path. Certain values of the process data are also presented with traced charts. These charts cover the wire-feeding rollers current and CMT device process Current and Voltage values. The advantage that these charts provide can be observed in Appendix E. CMT process report meters indicates that Current and Voltage values has dropped down to zero level for some part of the process. From the sequential pages viewer can indicate the timeline for this drop down. User is now eligible to access the history data Dashboard and zoom in for the questioned timeline of the process for studying the case in detail.

During the timetable when designing the reports within the thesis, the COAXwire process was not as well studied as CMT. Nevertheless, a report layout was designed to hold the basis and few detailed records. Most significant objective was to design a similar foundation corresponding with CMT report for future modifications. COAXwire finalized process report starts with a coversheet representing a timestamp and process name. First and only datasheet holds summary for finalized process as well as minimum, maximum and average values of the accessed laser power.

The layout of the report triggered at the start of a new manufacturing process, is slightly different. At the initial moment when new process is started only the basic information is available for accessing. For this reason, the layout was designed to hold all the information within the coversheet itself. Project team working with the environment can now get an immediate knowledge of an ongoing manufacturing test with just one look.

For manually triggering the report over the most recent process, a user can access a button element located at the Supervisory production monitoring Dashboard. Otherwise, the report triggering is done automatically. One additional page of the Supervisory Dashboard was harnessed to hold the logic for new report generation and triggering. Same logic could have been added to any of the already existing Dashboard pages. Yet, when adding one specific Dashboard page, a new updates and modifications of the logic are more easily comprehended. Boolean variable indicating the state of the environment, located at the *ProcessMonitor* device under *Production* path, was used as for triggering element for the report of a new started process. When triggering the finalized process report, a different approach was developed. Report can only be created and sent after all the data is stored inside the IoT-Ticket Data Server. Indicating the finalized data transfer a set of two new Boolean variables were added inside each *ProcessData* path, according the process title (CTM or COAXwire). These Boolean variables are updated at the start of the data transfer and together with the last line of the process data file. Change in the variable triggers a new report according the finalized process. For knowing the start and end time of the process, pair of variables are used from *ProcessMonitor - Production* path. These variables hold the timestamps with unix time millisecond values. Method of updating the variables is described in Chapter 4.4.2.

5. CONCLUSIONS

Following chapter concludes the work for the thesis. Keynotes for the work are detailed. Furthermore, results for the studied technology and when deployed in the field of production automation are discussed. Chapter closes with the future work consideration of the implementation.

5.1 Thesis conclusions

Thesis took a view for the paradigm change in the production automation and in manufacturing industries. Both individual customers and companies are requesting more customization for their products and at the same time, they are demanding better monitoring capabilities for their goods. In the past, manufacturing companies have formed clusters around the certain production artefacts. Now within these clusters the need for information exchange is rapidly evolving. Companies are furthermore shifting from capacity model for capability model.

Main focus of the thesis where kept in modern cloud computing technologies, resources, and Internet-of-Things technologies, providing the tools for companies to handle their process and production monitoring. In seems clear that by harnessing the capabilities within rapidly developing cloud and IoT technologies and combining these resources with Resource-Oriented- and Service-Oriented Architectures, new monitoring practises can be commissioned. With novel solutions both academic research and small-medium sized enterprises are not required for having extensive ICT skills to gain monitoring and reporting solutions. Cloud technology is additionally enhancing the computational power for deeper analysing of the gathered data.

In the thesis a modern Additive Manufacturing technology called Direct Energy Deposition was studied for the purpose of building the implementation for monitoring the process and the production within the Tampere University of Technology DED technology research cell. With the help of the results from the study and the experience working with the according research extensively, gave the tools for designing the implementation. Implemented solution monitors the research cell production and process in real-time, at the same time process data is gathered for later detailed reviewing by the user.

Implemented solution worked well in the tests and the review from the researches where mainly positive. Solution enables the monitoring of the application environment at the higher level, Supervisory level. This enables the monitoring via internet connection and an instant overview of the manufacturing process can be provided for the user. Gathered

process History Data can be studied online from individual users (researchers) workstations or homes or at any location equipped with available internet connection. This enhances the usage of the research cell. History Data can be reviewed without the need for working beside the environment where other teams are working in parallel.

It can be concluded that introduced and described technology suits well for the production automation and process monitoring applications. All studied capabilities open a great deal of opportunities for improving research, company's businesses, finding new businesses and providing additional features within company's products.

5.2 Future work

Future holds new possibilities for further developing of the implementation. User experience feedback from the research is constantly enhancing the interface of the IoT-Ticket Dashboards. New variables are added for gathering and monitoring of the process. Reports are taking them new shape as learning from the processes takes place. Additional sensors are planned to be installed for monitoring the process environment with more detail. One key feature gained when using IoT technology as one art of the solution. Room and process temperature with cooling water flow and temperature are ones that has been considered so far.

Information from the finished process should be taken inside PLM software for learning more details over the manufacturing with DED method. Such action improves the designing of the products manufactured with questioned manufacturing method. Life cycle management for the manufactured products and research cell itself can be harnessed. Devices monitored can be maintained with indications inserted via Dashboards. When combining all the possible aspects from the design of the products to the final production and finishing of the product and furthermore life cycle management capabilities both product itself and the environment and covering all these matters with modern cloud computing information systems, a Future Industry can be realized.

Due to the circumstances, that additive manufacturing research project was focusing on the two novel solution of CMT and COAXwire. Powder feeding device was left out from the data gathering and monitoring solution at this stage. Now when the platform exists adding the device is no massive effort to be conducted. When the project reaches the level where powder feeding as a manufacturing methods is included, the designed solution is updated to incorporate data gathering and monitoring.

Wapice has just now released a Mobile Designer tool. A new feature continuing their previous Interface Designer, Dataflow Editor and Report Editor tools. This new tool is intended for designing mobile first Dashboards. Tool comprehends a countless models of smart phone platforms for making the designs on specific model or study the suitabil-

ity of the designed Dashboard for other phone models. Mobile Designer tool is one near future features to be incorporated in the implementation.

As a separate study branch, the backend server implementation could be copied on different service provider's resources. This action would give a new hands-on experience over the steps that are required when shifting between different service providers. Another key future prospect is the study conducted by means of machining learning for the gathered data. Now when the data is stored in the cloud environment, adaptors can be used or designed, for further making analysis for improving the production with the help of studying patterns from the different processes. This data analysing together with additional sensor installations will be the next short term future objectives.

REFERENCES

[1] H. He, What is Service-Oriented Architecture, United International College, Honk Kong, web page. Available (accessed 4.9.2016): http://uic.edu.hk/~spjeong/ete/xml_what_is_service_oriented_architecture_sep2003.pdf.

[2] J. Salo, Designing a RESTful grid computing system : Master of Science Thesis, Tampere University of Technology, Tampere, 2010, vi, 54 p.

[3] Introduction to Cloud Computing, Dialogic, web page. Available (accessed 4.9.2016): https://www.dialogic.com/~/media/products/docs/whitepapers/12023-cloud-computing-wp.pdf.

[4] S. Almulla, Y. Iraqi, S. D. Wolthusen, Inferring relevance and presence of evidence in service-oriented and SaaS architectures, 2015 IEEE Symposium on Computers and Communication (ISCC), pp. 152-159.

[5] J. Juhanko, M. Jurvansuu, T. Ahlqvist, H. Ailisto, P. Alahuhta, J. Collin, M. Halen, T. Heikkilä, H. Kortelainen, M. Mäntylä, T. Seppälä, M. Sallinen, M. Simons, A. Tuominen, Suomalainen teollinen internet - haasteesta mahdollisuudeksi, 42, ETLA, 2015, .

[6] O. Bello, S. Zeadally, Intelligent Device-to-Device Communication in the Internet of Things, IEEE Systems Journal, Vol. 10, No. 3, 2016, pp. 1172-1182.

[7] Z. Duan, Y. Cao, M. Song, A construction method and data migration strategy for hybrid cloud storage, 2015 18th International Conference on Computer and Information Technology (ICCIT), pp. 473-478.

[8] Lehto.T, Tehdas ei tietoa pihtaa, Tekniikka ja Talous, Vol. 57, No. 35, 2016, pp. 18-19.

[9] Frost & Sullivan, Future of Manufacturing in Europe, Market Insight Frost & Sullivan, USA, 2016.

[10] Frost & Sullivan, Applications on the Cloud - New Business Models, Technical Insight Report D6CE, Frost & Sullivan, USA, 2016.

[11] K.V. Wong, A. Hernandez, A review of additive manufacturing, ISRN Mechanical Engineering, Vol. 2012, 2012.

[12] D.D. Gu, W. Meiners, K. Wissenbach, R. Poprawe, Laser additive manufacturing of metallic components: materials, processes and mechanisms, International Materials Reviews, Vol. 57, No. 3, 2012, pp. 133-164.

[13] Metal Additive Manufacturing process, Inovar Communications Ltd, web page. Available (accessed 16.10.2016): http://www.metal-am.com/introduction-to-metal-additive-manufacturing-and-3d-printing/metal-additive-manufacturing-processes/.

[14] Directed Energy Deposition, Loughborough University, web page. Available (accessed 15.10.2016): http://www.lboro.ac.uk/research/amrg/about/the7categoriesofadditivemanufacturing/directedenergydeposition/.

[15] ABB IRB 4600, web page. Available (accessed 4.8.2016): http://www.iws.fraunhofer.de/en/business_fields/surface_treatment/laser_cladding/system_technology/COAXwire/_jcr_content/contentPar/sectioncomponent/sectioParsys/textblockwithpics/imageComponent1/image.img.large.jpg/1464252331243_coaxwire-400x600.jpg.

[16] ABB IRBP A-750, web page. Available (accessed 4.8.2016): http://www.iws.fraunhofer.de/en/business_fields/surface_treatment/laser_cladding/system_technology/COAXwire/_jcr_content/contentPar/sectioncomponent/sectioParsys/textblockwithpics/imageComponent1/image.img.large.jpg/1464252331243_coaxwire-400x600.jpg.

[17] Absolute Accuracy - Industrial Robot Option, ABB, Apr. 2011, Available: https://library.e.abb.com/public/e69d8dd25cd7d36bc125794400374679/AbsAccPR10072EN_R6.pdf.

[18] Prakasham.G, Krishna.L, Kumar.A, A Review on the Effect of Various Process Parameters in Cold Metal Transfer (CMT) GMAW Welding, in: International Journal of Engineering Research, ITMAE, 2016, pp. 432-435.

[19] Cold Metal Transfer - The Technology, Fronius International GmbH, Feb. 2014, Available: http://www.fronius.com/cps/rde/xbcr/SID-F2B6125B-5A5D560B/fronius_international/M_06_0001_EN_CMT_leaflet_44211_snapshot.pdf.

[20] Metal powders - the raw material, Inovar Communications Ltd, web page. Available (accessed 17.10.2016): http://www.metal-am.com/introduction-to-metal- additive-manufacturing-and-3d-printing/metal-powders-the-raw-materials/.

[21] Coaxial laser wire cladding head COAXwire, Fraunhofer-Gesellschaft, web page. Available (accessed 17.10.2016): http://www.iws.fraunhofer.de/en/business_fields/surface_treatment/laser_cladding/system_technology/COAXwire.html.

[22] Fraunhofer IWS, Fraunhofer-Gesellsschaft, web page. Available (accessed 17.10.2016): http://www.iws.fraunhofer.de/en.html.

[23] D. Fensel, F.M. Facca, E. Simperl, I. Toma, Semantic Web Services, 2011, XI, 357p p.

[24] S. M. Shariati, Abouzarjomehri, M. H. Ahmadzadegan, Challenges and security issues in cloud computing from two perspectives: Data security and privacy protection, 2015 2nd International Conference on Knowledge-Based Engineering and Innovation (KBEI), pp. 1078-1082.

[25] Visma Software, Pilvipalvelut yrityskäytössä, Visma, web page. Available (accessed 10.9.2016): http://www.visma.fi/tietopankki/opas/pilvipalvelut-yrityskaytossa/.

[26] A. Singh, S. Sharma, S. R. Kumar, S. A. Yadav, Overview of PaaS and SaaS and its application in cloud computing, 2016 International Conference on Innovation and Challenges in Cyber Security (ICICCS-INBUSH), pp. 172-176.

[27] Q. Li, R. Li, Reliability evaluation for cloud computing system considering common cause failure, 2016 35th Chinese Control Conference (CCC), pp. 5267-5271.

[28] H. Brabra, A. Mtibaa, L. Sliman, W. Gaaloul, F. Gargouri, Semantic Web Technologies in Cloud Computing: A Systematic Literature Review, 2016 IEEE International Conference on Services Computing (SCC), pp. 744-751.

[29] Tietotekniikan käyttö yrityksissä (2014), Statistics Finland, web page. Available (accessed 4.9.2016): http://www.stat.fi/til/icte/2014/icte_2014_2014-11-25_tie_001_fi.html.

[30] Tietotekniikan käyttö yrityksissä (2015), Statistics Finland, web page. Available (accessed 4.9.2016): http://www.stat.fi/til/icte/2015/icte_2015_2015-11-26_tie_001_fi.html.

[31] M. AhmadKhan, A survey of security issues for cloud computing, Journal of Network and Computer Applications, 2016, pp. 11-29.

[32] S. Yamamoto, S. Matsumoto, S. Saiki, M. Nakamura, Materialized View as a Service for Large-Scale House Log in Smart City, 2013 IEEE 5th International Conference on Cloud Computing Technology and Science, pp. 311-316.

[33] Y. Benslimane, Z. Yang, B. Bahli, Key Topics in Cloud Computing Security: A Systematic Literature Review, Information Science and Security (ICISS), 2015 2nd International Conference on, pp. 1-4.

[34] S. Narula, A. Jain, Prachi, Cloud Computing Security: Amazon Web Service, 2015 Fifth International Conference on Advanced Computing & Communication Technologies, pp. 501-505.

[35] Patil Madhubala R., Survey on security concerns in Cloud computing, Green Computing and Internet of Things (ICGCIoT), 2015 International Conference on, pp. 1458-1462.

[36] R. Merrit, New group aims to secure PCs, PDAs, cell phones, EE-Times, web page. Available (accessed 18.9.2016): http://www.eetimes.com/document.asp?doc_id=1202119.

[37] R. Kaur, J. Kaur, Cloud computing security issues and its solution: A review, Computing for Sustainable Global Development (INDIACom), 2015 2nd Inter national Conference on, pp. 1198-1200.

[38] N. Zhang, R. Li, Resource optimization with reliability consideration in cloud computing, 2016 Annual Reliability and Maintainability Symposium (RAMS), pp. 1-6.

[39] J. Soldatos, S. Gusmeroli, G. Di Orio, Internet of Things Applications in Future Manufacturing, in: O. Vermesan, P. Friess (ed.), Digitising the Industry Internet of Things Connecting the Physical, Digital and Virtual Worlds, Volume 49 ed., River Publishers, Denmark, 2016, pp. 153-182.

[40] K.Peltomaa, Sales manager, Pronius Oy, Tampere, Interview 9.6.2016.

[41] ABB, Robot Web Services, ABB, web page. Available (accessed 22.9.2016): http://developercenter.robotstudio.com/Index.aspx?DevCenter=Robot_Web_Services&OpenDocument&Url=html/index.html.

[42] V. C. Emeakaroha, N. Cafferkey, P. Healy, J. P. Morrison, A Cloud-Based IoT Data Gathering and Processing Platform, Future Internet of Things and Cloud (FiCloud), 2015 3rd International Conference on, pp. 50-57.

[43] M. C. Domenech, L. P. Rauta, M. D. Lopes, P. H. Da Silva, R. C. Da Silva, B. W. Mezger, M. S. Wangham, Providing a Smart Industrial Environment with the Web of Things and Cloud Computing, 2016 IEEE International Conference on Services Computing (SCC), pp. 641-648.

[44] E. Tragos, J. Bernabe, R. Staudemeyer, J. Ramos, A. Fragkiadakis, A. Skarmeta, M. Nati, A. Gluhak, Trusted IoT in the Complex Landscape of Governance, Security, Privacy, Availability and Safety, in: O. Vermesan, P. Friess (ed.), Digitising the Industry Internet of Things Connecting the Physical, Digital and Virtual Worlds, Volume 49 ed., River Publishers, Denmark, 2016, pp. 185-210.

[45] O. Vermesan, P. Friess, P. Guillemin, M. Serrano, M. Bouraoui, L. Freire, T. Kallstenius, K. Lam, M. Eisenhauer, K. Moessner, M. Spirito, E. Tragos, H. Sundmaeker, P. Malo, A. Wees, IoT Digital Value Chain Connecting Research, Innovation and Deploymenty, in: O. Vermesan, P. Friess (ed.), Digitising the Industry Internet of Things Connecting the Physical, Digital and Virtual Worlds, Volume 49 ed., River Publishers, Denmark, 2016, pp. 15-129.

[46] Pro ADO.NET Data Services : Working with RESTful Data, in: Springer eBooks, Apress, Computer science.Computer ScienceProgramming Techniques, Berkeley, CA, 2009, .

[47] E. Wilde, C. Pautasso, REST: From Research to Practice, 2011, XXX, 496p. 150 illus p.

[48] R.T. Fielding, Architectural styles and the design of network-based software architectures, Dissertation, University of California, Irvine, 2000, 180 p.

[49] L. Richardson, S. Ruby, RESTful web services, First Edition ed. O'Reilly Media, Inc., USA, Sebastopol, 2007, 416 p.

[50] C. Pautasso, E. Wilde, R. Alarcon, REST: Advanced Research Topics and Practical Applications, 2014, IX, 222 pages 58 illus., 25 illus. in color p.

[51] P. Adamczyk, P. Smith, R. Johnson, M. Hafiz, REST and Web Services: In Theory and in Practice, in: E. Wilde, C. Pautasso (ed.), REST: From Research to Practice, Springer, USA, 2011, pp. 35-57.

[52] HTTP Status Codes, RestApiTutorial, web page. Available (accessed 7.10.2016): http://www.restapitutorial.com/httpstatuscodes.html.

[53] N. Mäkitalo, REST-pohjainen sisällönhallintajärjestelmä hajautettuun ympäris töön : diplomityö, Tampereen teknillinen yliopisto, Tampere, 2011, vi, 81 lehteä p.

[54] HTTP Methods, RestApiTutorial, web page. Available (accessed 8.10.2016): http://www.restapitutorial.com/lessons/httpmethods.html.

[55] U. Thakar, A. Tiwari, S. Varma, On Composition of SOAP Based and RESTful Services, 2016 IEEE 6th International Conference on Advanced Computing (IACC), pp. 500-505.

[56] R. Verborgh, A. Harth, M. Maleshkova, S. Stadtmüller, T. Steiner, M. Taheriyan, R. Van de Walle, Survey of Semantic Description of REST APIs, in: C. Pautasso, E. Wilde, R. Alarcon (ed.), REST: Advanced Research Topics and Practical Applications, Springer, USA, 2014, pp. 69-89.

[57] FTP, TechTerms, web page. Available (accessed 19.9.2016): http://techterms.com/definition/ftp.

[58] Secure Shell, Techtarget, web page. Available (accessed 19.9.2016): http://searchsecurity.techtarget.com/definition/Secure-Shell.

[59] Use of vsftp for secure, reliable FTP server, Techrepublic, web page. Available (accessed 19.9.2016): http://www.techrepublic.com/article/use-vsftp-for-a-secure-reliable-ftp-server/.

[60] What is File Transfer Protocol (FTP?) - Definition & Explanation, study.com, web page. Available (accessed 19.9.2016): http://study.com/academy/lesson/what-is-file-transfer-protocol-ftp-definition-lesson-quiz.html.

[61] About AWS, Amazon WEb Services, web page. Available (accessed 18.10.2016): https://aws.amazon.com/about-aws/.

[62] Amazon Global Infrastructure, Amazon Web Services, web page. Available (accessed 18.10.2016): https://aws.amazon.com/about-aws/global-infrastructure/.

[63] Magic Quadrant for Cloud Infrastructure as a Service, Worldwide, Gartner Inc., web page. Available (accessed 18.10.2016): https://www.gartner.com/doc/reprints?id=1-2G2O5FC&ct=150519.

[64] AWS Free Tier, Amazon WEb Services, web page. Available (accessed 18.10.2016): https://aws.amazon.com/free/.

[65] AWS Cloud Security, Amazon WEb Services, web page. Available (accessed 18.10.2016): https://aws.amazon.com/security/.

[66] AWS Customer Agreement, Amazon WEb Services, web page. Available (accessed 18.10.2016): https://aws.amazon.com/agreement/.

[67] Amazon EC2 - Virtual Server Hosting, Amazon WEb Services, web page. Available (accessed 18.10.2016): https://aws.amazon.com/ec2/.

[68] Amazon ec2 LAMP and FTP installation and setup, Github, web page. Available (accessed 18.10.2016): https://gist.github.com/gunjanpatel/37d306cd1585ece1179b.

[69] S3 FTP: build a reliable and inexpensive FTP server using Amazon's S3, CloudAcademy, web page. Available (accessed 18.10.2016): http://cloudacademy.com/blog/s3-ftp-server/.

[70] Setting up FTP on Amazon Cloud Server, Stackoverflow, web page. Available (accessed 18.10.2016): http://stackoverflow.com/questions/7052875/setting-up-ftp-on-amazon-cloud-server.

[71] How to install nodejs 4.x in AWS EC2 instance, Easyusedev, web page. Available (accessed 18.10.2016): https://easyusedev.wordpress.com/2016/02/04/how- to-install-node-js-4-x-in-aws-ec2-instance/.

[72] AWS IoT, Amazon Web Services, web page. Available (accessed 18.10.2016): https://aws.amazon.com/iot/?nc2=h_m1.

[73] Amazon Relational Database Service (RDS), Amazon Web Services, web page. Available (accessed 18.10.2016): https://aws.amazon.com/rds/.

[74] AWS Quicksight, Amazon Web Services, web page. Available (accessed 18.10.2016): https://aws.amazon.com/quicksight/.

[75] AWS Partner Directory, Amazon Web Services, web page. Available (accessed 18.10.2016): http://www.aws-partner-directory.com/PartnerDirectory/.

[76] Upcoming name change for Windows Azure, 2014 Microsoft, web page. Available (accessed 19.10.2016): https://azure.microsoft.com/en-us/blog/upcoming-name-change-for-windows-azure/.

[77] A Brief History of Microsoft Azure, Gary Clarke, web page. Available (accessed 19.10.2016): http://garyclarke.us/technology/a-brief-history-of-microsoft-azure/.

[78] Azure Regions, 2016 Microsoft, web page. Available (accessed 19.10.2016): https://azure.microsoft.com/en-us/regions/.

[79] Azure Products, 2016 Microsoft, web page. Available (accessed 19.10.2016): https://azure.microsoft.com/en-us/services/?sort=popular&filter=all.

[80] Get started with Azure, 2016 Microsoft, web page. Available (accessed 19.10.2016): https://azure.microsoft.com/en-us/get-started/.

[81] Azure Pricing Calculator, 2016 Microsoft, web page. Available (accessed 19.10.2016): https://azure.microsoft.com/en-us/pricing/calculator/.

[82] Azure Technology and Service Partners, 2016 Microsoft, web page. Available (accessed 19.10.2016): https://azure.microsoft.com/en-us/partners/directory/.

[83] Azure Customer case studies, 2016 Microsoft, web page. Available (accessed 19.10.2016): https://azure.microsoft.com/en-us/case-studies/.

[84] Trusted Cloud: Windows Azure-Security, Privacy and Compliance, Microsoft, Apr. 2015, Available: http://download.microsoft.com/download/1/6/0/160216AA-8445-480B-B60F-5C8EC8067FCA/WindowsAzure-SecurityPrivacyCompliance.pdf.

[85] Create Azure Virtual Machine and Setup Node.js, 2016 Girnar Software Pvt.Ltd, web page. Available (accessed 19.10.2016): http://www.connecto.io/blog/create-a-node-js-server-on-azure-vm-with-mongodb-as-database-in-added-vhd/.

[86] How to configure FTP on Azure Linux VM, 2016 Code Chewing, web page. Available (accessed 19.10.2016): http://www.codechewing.com/library/configure-ftp-azure-linux-ubuntu-vm/.

[87] Overview Azure IoT Hub, 2016 Microsoft, web page. Available (accessed 19.10.2016): https://azure.microsoft.com/en-us/documentation/videos/azurecon-2015-overview-of-azure-iot-hub/.

[88] Get started with Azure IoT Hub, 2016 Microsoft, web page. Available (accessed 19.10.2016): https://azure.microsoft.com/en-us/documentation/articles/iot-hub-node-node-getstarted/.

[89] What is Azure IoT Suite? 2016 Microsoft, web page. Available (accessed 19.10.2016): https://azure.microsoft.com/en-us/documentation/articles/iot-suite-overview/.

[90] Azure Blob Storage, 2016 Microsoft, web page. Available (accessed 19.10.2016): https://azure.microsoft.com/en-us/services/storage/blobs/.

[91] Microsoft Power BI, 2016 Microsoft, web page. Available (accessed 19.10.2016): https://powerbi.microsoft.com/en-us/.

[92] Azure Power BI Embedded, 2016 Microsoft, web page. Available (accessed 19.10.2016): https://azure.microsoft.com/en-us/services/power-bi-embedded/.

[93] Google App Engine Release, Google Inc., web page. Available (accessed 20.10.2016): http://googleappengine.blogspot.fi/2008/04/introducing-google-app-engine-our-new.html.

[94] Google Regions and Zones, Google Inc., web page. Available (accessed 20.10.2016): https://cloud.google.com/compute/docs/regions-zones/regions-zones.

[95] Google Cloud Platform Products, Google Inc., web page. Available (accessed 20.10.2016): https://cloud.google.com/products/.

[96] Google Cloud Platform Pricing, Google Inc., web page. Available (accessed 20.10.2016): https://cloud.google.com/pricing/tml.

[97] Building node.js application on GCP, Google Cloud 2016, web page. Available (accessed 20.10.2016): https://www.youtube.com/watch?v=jsznS0QxtYI.

[98] Google - How does the free trial work? Google Inc., web page. Available (accessed 20.10.2016): https://cloud.google.com/free-trial/docs/.

[99] Google Cloud Platform Pricing Calculator, Google Inc., web page. Available (accessed 20.10.2016): https://cloud.google.com/products/calculator/#id=60130062-3df6-48ca-9e5b-933ee41b4bd7.

[100] Google Cloud Platform Security Whitepaper, Google Inc., web page. Available (accessed 20.10.2016): https://cloud.google.com/security/whitepaper.

[101] Google Cloud Platform Term of Service, Google Inc., web page. Available (accessed 20.10.2016): https://cloud.google.com/terms/.

[102] Google Cloud Platform Service Specific Terms, Google Inc., web page. Available (accessed 20.10.2016): https://cloud.google.com/terms/service-terms.

[103] Google Cloud Platform Why to choose? Google Inc., web page. Available (accessed 20.10.2016): https://cloud.google.com/why-google/.

[104] Google Cloud Platform Compute Engine, Google Inc., web page. Available (accessed 20.10.2016): https://cloud.google.com/compute/.

[105] Running Node.js on Compute Engine, Google Inc., web page. Available (accessed 20.10.2016): https://cloud.google.com/nodejs/getting-started/run-on-compute-engine.

[106] Google Cloud Platform Cloud Storage, Google Inc., web page. Available (accessed 20.10.2016): https://cloud.google.com/storage/.

[107] Google Cloud Platform Cloud SQL, Google Inc., web page. Available (accessed 20.10.2016): https://cloud.google.com/sql/.

[108] Google Cloud Platform Cloud Datalab, Google Inc., web page. Available (accessed 20.10.2016): https://cloud.google.com/datalab/.

[109] Jupyter Notebook, 2016 Project Jupyter, web page. Available (accessed 20.10.2016): http://jupyter.org/.

[110] About SoftLayer, SoftLayer Technologies Inc., web page. Available (accessed 18.10.2016): http://www.softlayer.com/about-softlayer.

[111] Rackspace - Cloud expertise, Rackspace, web page. Available (accessed 18.10.2016): http://www.rackspace.co.uk/cloud-expertise.

[112] CenturyLink - Managed office, CenturyLink, web page. Available (accessed 18.10.2016): http://www.centurylink.com/business/.

[113] Virtustream - The cloud for the core of the enterprise, Virtustream, web page. Available (accessed 18.10.2016): http://www.virtustream.com/solutions/mission-critical-apps.

[114] UpCloud features, UpCloud, web page. Available (accessed 18.10.2016): https://www.upcloud.com/fi/ominaisuudet/.

[115] S. Yadav, Comparative study on open source software for cloud computing platform: Eucalyptus, OpenStack and OpenNebula, International Journal Of Engineering And Science, Vol. 3, No. 10, 2013, pp. 51-54.

[116] Wapice Company, Wapice Ltd., web page. Available (accessed 12.10.2016): https://www.wapice.com/about-us/wapice.

[117] IoT-Ticket, Wapice Ltd., web page. Available (accessed 12.10.2016): https://www.iot-ticket.com/platform.

[118] Freeboard.io, Bug Labs Inc, web page. Available (accessed 9.10.2016): https://freeboard.io/.

[119] Ignition Discover the SCADA, Inductive Automation, web page. Available (accessed 10.10.2016): https://inductiveautomation.com/scada-software/#.

[120] Ignition IIoT, Inductive Automation, web page. Available (accessed 10.10.2016): https://inductiveautomation.com/ignition-iiot.

[121] Ignition Software pricing, Inductive Automation, web page. Available (accessed 10.10.2016): https://inductiveautomation.com/pricing/ignition.

[122] DGlogik, DGlogik Inc, web page. Available (accessed 10.10.2016): http://www.dglogik.com/company.

[123] DGlogik IoE Application Platform, DGlogik Inc, web page. Available (accessed 10.10.2016): http://www.dglogik.com/products/dglux5-ioe-application-platform.

[124] DGlogik IoT Verticas, DGlogik Inc, web page. Available (accessed 10.10.2016): http://www.dglogik.com/iot-verticals.

[125] Introduction to DGLux5, DGLogik, Available: http://www.dglogik.com/component/phocadownload/category/1-dglogik- marketing-material?download=1:introduction-to-dglux5.

[126] Infinite Automation: DGLux5 Pricing, Infinite Automation, 2016, Available: http://infiniteautomation.com/forum/uploads/files/1452182616305-dglux5-price-list.pdf.

[127] Cascading Style Sheets, W3Schools, web page. Available (accessed 11.10.2016): http://www.w3schools.com/css/css_intro.asp.

[128] Yeoman generator ecosystem, Yeoman, web page. Available (accessed 11.10.2016): http://yeoman.io/.

[129] Sass CSS, Sass Hampton Catlin, Natalie Weizenbaum, Chris Eppstein, and numerous contributors, web page. Available (accessed 11.10.2016): http://sass-lang.com/.

[130] Bootstrap framework, Bootstrap Core Team, web page. Available (accessed 11.10.2016): http://getbootstrap.com/.

[131] jQuery, jQuery Foundation, web page. Available (accessed 11.10.2016): https://jquery.com/.

[132] T. Latvala, Deployment of a service-oriented automation platform for integrating smart city applications : Master of Science Thesis, Tampere University of Technology, Tampere, 2010, vi, 58 p.

[133] The 14 best data visualization tools, The NextWeb, web page. Available (accessed 11.10.2016): http://thenextweb.com/dd/2015/04/21/the-14-best-data-visualization-tools/#gref.

[134] JavaScript General Introduction, Quirksmode, web page. Available (accessed 8.10.2016): http://www.quirksmode.org/js/intro.html.

[135] nodeJS, NodeJS Foundation, web page. Available (accessed 9.10.2016): https://nodejs.org/en/.

[136] Fronius: CMT Power Source, Fronius, web page. Available (accessed 21.10.2016): http://www.fronius.com/internet/img/w_v_13_FE1_8.jpg.

[137] ARM Fibre laser, Corelase Oy, 2015, Available: http://www.corelase.fi/wp-content/uploads/2016/03/arm_A4_4s_0906.pdf.

[138] Fraunhofer: Surface Treatment, Fraunhofer, web page. Available (accessed 4.8.2016): http://www.iws.fraunhofer.de/en/business_fields/surface_treatment/laser_cladding/system_technology/COAXwire/_jcr_content/contentPar/sectioncomponent/sectioParsys/textblockwithpics/imageComponent1/image.img.large.jpg/1464252331243_coaxwire-400x600.jpg.

[139] Fronius Xplorer, Pronius Oy, web page. Available (accessed 15.10.2016): http://www.pronius.fi/?p=114.

[140] AWS CloudWatch, Amazon Web Services, web page. Available (accessed 18.10.2016): https://aws.amazon.com/cloudwatch/details/.

[141] AWS What is Amazon VPC? Amazon Web Services, web page. Available (accessed 25.10.2016): http://docs.aws.amazon.com/AmazonVPC/latest/UserGuide/VPC_Introduction.html.

[142] AWS VPC Endpoints, Amazon Web Services, web page. Available (accessed 22.10.2016): http://docs.aws.amazon.com/AmazonVPC/latest/UserGuide/vpc-endpoints.html#vpc-endpoints-s3-bucket-policies.

[143] AWS Your VPC and subnets, Amazon Web Services, web page. Available (accessed 25.10.2016): http://docs.aws.amazon.com/AmazonVPC/latest/UserGuide/VPC_Subnets.html#SubnetSecurity.

[144] CIDR (Classless Inter-Domain Routing or supernetting), TechTarget, web page. Available (accessed 25.10.2016): http://searchnetworking.techtarget.com/definition/CIDR.

[145] RFC 1918, Internet Engineering Task Force, web page. Available (accessed 25.10.2016): https://tools.ietf.org/html/rfc1918.

[146] AWS Internet Gateways, Amazon Web Services, web page. Available (accessed 25.10.2016): http://docs.aws.amazon.com/AmazonVPC/latest/UserGuide/VPC_Internet_Gateway.html.

[147] AWS Elastic IP addresses, Amazon Web Services, web page. Available (accessed 25.10.2016): http://docs.aws.amazon.com/AWSEC2/latest/UserGuide/elastic-ip-addresses-eip.html.

[148] AWS Security in Your VPC, Amazon Web Services, web page. Available (accessed 25.10.2016): http://docs.aws.amazon.com/AmazonVPC/latest/UserGuide/VPC_Security.html.

[149] AWS Elastic Beanstalk, Amazon Web Services, web page. Available (accessed 26.10.2016): https://aws.amazon.com/elasticbeanstalk/.

[150] AWS EC2 Product Details, Amazon Web Services, web page. Available (accessed 25.10.2016): https://aws.amazon.com/ec2/details/.

[151] AWS EC2 Instance Types, Amazon Web Services, web page. Available (accessed 26.10.2016): https://aws.amazon.com/ec2/instance-types/.

[152] AWS EC2 Security Groups for Linux Instances, Amazon Web Services, web page. Available (accessed 26.10.2016): http://docs.aws.amazon.com/AWSEC2/latest/UserGuide/using-network-security.html#ec2-classic-security-groups.

[153] AWS Connecting to Your Linux instance from Windows Using PuTTY, Amazon Web Services, web page. Available (accessed 26.10.2016): http://docs.aws.amazon.com/AWSEC2/latest/UserGuide/putty.html.

[154] PuTTY, PuTTY.org, web page. Available (accessed 26.10.2016): http://www.putty.org/.

[155] AWS Amazon Machine Images, Amazon Web Services, web page. Available (accessed 26.10.2016): http://docs.aws.amazon.com/AWSEC2/latest/UserGuide/AMIs.html.

[156] AWS S3 Product Details, Amazon WEb Services, web page. Available (accessed 26.10.2016): https://aws.amazon.com/s3/details/.

[157] AWS SDK API S3, Amazon WEb Services, web page. Available (accessed 27.10.2016): http://docs.aws.amazon.com/AWSJavaScriptSDK/latest/AWS/S3.html.

[158] AWS What is IAM? Amazon Web Services, web page. Available (accessed 27.10.2016): http://docs.aws.amazon.com/IAM/latest/UserGuide/introduction.html.

[159] I.M. Delamer, J.L. Martinez Lastra, Factory information systems in electronic production, Tampere University of Technology, Tampere, 2007, 293 sivua p.

[160] AWS Amazon RDS Product Details, Amazon Web Services, web page. Available (accessed 27.10.2016): https://aws.amazon.com/rds/details/.

[161] AWS Using Amazon RDS with Node.js, Amazon Web Services, web page. Available (accessed 27.10.2016): http://docs.aws.amazon.com/elasticbeanstalk/latest/dg/create_deploy_nodejs.rds.html.

[162] MySQL Community Edition, Amazon Web Services, web page. Available (accessed 27.10.2016): https://www.mysql.com/products/community/

[163] SQL Tutorial, W3Schools.com, web page. Available (accessed 27.10.2016): http://www.w3schools.com/sql/.

[164] Latest IoT-Ticket News, Wapice Ltd., web page. Available (accessed 29.10.2016): https://www.iot-ticket.com/news.

[165] IOT API Documentation, IoT-Ticket.com REST API, Wapice Ltd, Vaasa, 2016, 29 p.

[166] IoT-Ticket Marketplace, Wapice Ltd., web page. Available (accessed 30.10.2016): https://www.iot-ticket.com/marketplace.

[167] RFC 2069, Internet Engineering Task Force, web page. Available (accessed 27.2.2017): https://tools.ietf.org/html/rfc2069.

[168] RFC 2617, Internet Engineering Task Force, web page. Available (accessed 27.2.2017): https://tools.ietf.org/html/rfc2617.

[169] O.Hakaluoto, Founder, Roboco Oy, Helsinki, Interview 24.11.2016.

[170] IP address space calculator, Jodies.de, web page. Available (accessed 31.12.2016): http://jodies.de/ipcalc?host=172.32.0.0&mask1=20&mask2=.

[171] log4js - Loggly module, 2017 Github, Inc, web page. Available (accessed 12.02.2017): https://github.com/nomiddlename/log4js-node/wiki/Loggly.

[172] Loggly - Intro to Log Management, 2017 Loggly Inc, web page. Available (accessed 12.02.2017): https://www.loggly.com/intro-to-log-management/.

APPENDIX A: IOT-TICKET REQUEST-RESPONSE MESSAGES

Request-Response message structures for registering a device [165]

Request: https://{api-server-URL}/devices/	
Parameter	**Description**
name (String:100) (mandatory field)	Name of the device
manufacturer (String:100) (mandatory field)	name of the device manufacturer
type (String:100)	Category of the device
description (String:100)	Description over the device
attributes (key,String:255)(value,String:255)	Array of key value pairs, for storing additional attributes

Response:	
Field	**Description**
href	URL for accessing the resource
deviceID	Device ID for subsequent calls. Consisting of 32 alphanumeric characters.
name	Name provided at the registration message
manufacturer	Manufacturer provided at the registration message
type	Type provided at the registration message
createdAt	Time of the creation. ISO8601 UTC format.
attributes	Attributes provided at the registration message
description	Description of the device

Request-Response message structures for devices listings [165]

Request: https://{api-server-URL}/devices/		
Parameter	**Description**	**Examples**
callback	JavaScript function triggered at response	/devices?callback=bar returns: bar(response data)
format	Format /JSON / XML)	/devices?format=xml
limit	Number of returned results. Max 100, default 10	/devices?limit=5
offset	Number results to be skipped from the beginning	/devices?offset=3

Response:	
Field	**Description**
offset	Number of skipped results from the start
limit	Results per one page
fullsize	Total amount of applicaple devices for client
devices	*List of the objects with following parameters:*
href	URL to the device
name	Name provided at the registration message
manufacturer	Manufacturer provided at the registration message
type	Type provided at the registration message
createdAt	Time of the creation. ISO8601 UTC format.
description	Description of the device
attributes	Attributes provided at the registration message

Request-Response message structures for a device information [165]

Request: https://{api-server-URL}/devices/deviceID		
Parameter	**Description**	**Examples**
callback	JavaScript function triggered at response	/devices/deviceID?callback=bar returns: bar(response data)
format	Format (JSON / XML)	/devices/deviceID?format=xml

Response:	
Field	**Description**
href	URL to the device
name	Name provided at the registration message
type	Type provided at the registration message
manufacturer	Manufacturer provided at the registration message
createdAt	Time of the creation. ISO8601 UTC format.
description	Description of the device
attributes	Attributes provided at the registration message

Request-Response message structures for a device datanodes listing [165]

Request: https://{api-server-URL}/devices/deviceID/datanodes		
Parameter	**Description**	**Examples**
callback	JavaScript function triggered at response	/devices/deviceID/datanodes?callback=bar returns: bar(response data)
limit	Number of returned results. Max 100, default 10	/devices/deviceID/datanodes?limit=5
offset	Number results to be skipped from the beginning	/devices/deviceID/datanodes?offset=3
format	Format (JSON / XML)	/devices/deviceID/datanodes?format=xml

Response:	
Field	**Description**
offset	Number of skipped results from the start
limit	Results per one page
fullsize	Total amount of applicable data nodes for the device
items	List of data node objects

Request-Response message structure for writing data [165]

Request: https://{api-server-URL}/process/write/deviceID		
Parameter	**Description**	**Examples**
deviceID	ID of the target device	
Data to be written	*Array of data objects. Each of the objects holds the following attributes:*	
Name: (String:100)	Name of the data node	
Path: (String:100)	Forward slash separated list of path to be created for current data node. Max 10 components.	/Engine/LeftBlock/
v	Value to be written	
unit (String:10)	Unit of the written data	V, A, Hz
ts (long)	Timestamp according to Unix Timestamp. Milliseconds after the Epoch	
dataType	Data type of the variable	long / double / Boolean / string / binary

Response:	
Field	**Description**
write result object	Array of response objects each containing: • href • WrittenCount
totalWritten	Number of total data points written
href	URL for the current data node
WrittenCount	Values written to the data node

Request-Response message structure for reading data [165]

Request: https://{api-server-URL}/process/read/deviceID		
Parameter	**Description**	**Examples**
format	Format (JSON / XML)	/process/read/deviceID?datanodes =Temperature?format=xml
deviceID	Targetted deviceID	
datanodes	Comma separated list of the data nodes to be read. Number of requested data nodes should not exceed 10	/process/read/deviceID?datanodes =Temperature,AirIntake
fromdate (long)	Unix Timestamp. Time from which the data is requested	/process/read/deviceID?datanodes =Temperature?fromdate =1415260152284
todate (long)	Unix Timestamp. Time to which the data is requested	/process/read/deviceID?datanodes =Temperature?fromdate =1415260152284&todate =1417609677
limit (integer)	Max number of data points returned for each data node. Default value 1000, max value 10 000	/process/read/deviceID?datanodes =Temperature?limit=5
order	Order of the requested values by timestamp	/process/read/deviceID?datanodes =Temperature?order=descending

Response:	
Field	**Description**
objects	Array of response objects each containing: • href • type • unit • value with arrays of v and ts
name	Name of the data node
path	Path to get to data node
v	Value at the timestamp of ts
ts	Timestamp accomplice with the value
unit	Unit of the data node

APPENDIX B: REAL-TIME PROCESS MONITORING URI'S

Robot variable URI's for monitoring the CMT process

base-URI: http://robot-ip-address/rw/			
Name	**resource-URI**	**Datatype (IoT Ticket)**	**Unit (IoT Ticket)**
cmtJobNumber	/iosystem/signals/EtherNetIP/ioFronius1/goFr1JobNum?json=1	long	num
cmtCurrent	/iosystem/signals/EtherNetIP/ioFronius1/aiFr1Current_M?json=1	double	A
cmtVoltage	/iosystem/signals/EtherNetIP/ioFronius1/aiFr1Volt_M?json=1	double	V
SpeedRobot	/iosystem/signals/aoR1TCPSpeed?json=1	double	mm/s
weavingFreg	/iosystem/signals/aoR1TCPSpeed?json=1	double	Hz
torchAngle	/rapid/symbol/data/RAPID/T_ROB1/CellData/nTorchAngle?json=1	double	deg
arcLenghtCorr	/iosystem/signals/EtherNetIP/ioFronius1/aoFr1ArcLength?json=1	double	%
wirefeed_mms	/iosystem/signals/EtherNetIP/ioFronius1/aiFr1WireFeed_M?json=1	double	mm/s
wirefeed_perc	/iosystem/signals/EtherNetIP/ioFronius1/aoFr1Power?json=1	double	%
dynamicCorre	/iosystem/signals/EtherNetIP/ioFronius1/aoFr1Dynamic?json=1	double	num
wireFeedRollCurrent	/iosystem/signals/EtherNetIP/ioFronius1/aiFr1MotorCurr_M?json=1	double	A
MainCurrent	/iosystem/signals/EtherNetIP/ioFronius1/diFr1MainCurrent?json=1	Boolean	bool
Communication-Ready	/iosystem/signals/EtherNetIP/ioFronius1/diFr1CommunicRdy?json=1	Boolean	bool
SetArcON	/iosystem/signals/EtherNetIP/ioFronius1/doFr1ArcOn?json=1	Boolean	bool
ProcessActive	/iosystem/signals/EtherNetIP/ioFronius1/diFr1ProcessActv?json=1	Boolean	bool

Robot variable URI's for monitoring the COAXwire process

base-URI: http://robot-ip-address/rw/iosystem/signals			
Name	**resource-URI**	**Datatype (IoT Ticket)**	**Unit (IoT Ticket)**
laserPower	/EtherNetIP/Controller_IO/aoLaserPower?json=1	double	W
laserAlarm	/EtherNetIP/Controller_IO/diLaserAlarm?json=1	Boolean	bool
laserEmissionON	/EtherNetIP/Controller_IO/diLaserOn?json=1	Boolean	bool
laserReady	/EtherNetIP/Controller_IO/diLaserReady?json=1	Boolean	bool
laserAimingBeam	/EtherNetIP/Controller_IO/doLaserAiming?json=1	Boolean	bool
laserSetBeamON	/EtherNetIP/Controller_IO/doLaserBeamOn?json=1	Boolean	bool
laserClearAlarms	/EtherNetIP/Controller_IO/doLaserClearAlarms?json=1	Boolean	bool
coaxReady	/EtherNetIP/ioWirefeed/diWf1_Ready?json=1	Boolean	bool
coaxFault	/EtherNetIP/ioWirefeed/diWf2_Fault?json=1	Boolean	bool
coaxWireFault	/EtherNetIP/ioWirefeed/diWf3_WireFault?json=1	Boolean	bool
coaxLifeBit	/EtherNetIP/ioWirefeed/diWf7_LifeBit?json=1	Boolean	bool
coaxEmStop	/EtherNetIP/ioWirefeed/diWf11_EmStop?json=1	Boolean	bool
coaxProcActiv	/EtherNetIP/ioWirefeed/diWf14_ProcActiv?json=1	Boolean	bool
coaxWireEnd	/EtherNetIP/ioWirefeed/diWf15_WireEnd?json=1	Boolean	bool
coaxStartRelease	/EtherNetIP/ioWirefeed/doWf1_StartRelease?json=1	Boolean	bool
coaxStartFeed	/EtherNetIP/ioWirefeed/doWf3_StartFeed?json=1	Boolean	bool
coaxwActual-WireSpeed	/EtherNetIP/ioWirefeed/giWf_WirefeedSpeed?json=1	double	mm/s

APPENDIX C: SEGMENTS OF THE PROCESS DATA FILE

First and the last line of the process data file

Record name	Record definition
First line of the process data file	
Build platform	Identification of the process run, formed over date and time of the start moment
Process	CMT or COAXwire
Product	Fixture of the positioner for this particular process run
Details	Additional details for the process run
Used ID	Name of the user
Start	Indication that process is now started
Start time	Timestamp of the process start
Data frequency	Frequency by which the data is collected and stored
Last line of the process data file	
End	Indication that process is finished
End time	Timestamp of the finished process

Gathered data for CMT process (middle segment)

Record name	Record definition
Timestamp	Timestamp from the start of the process
Job number	Number of CMT job, welding parameters and synergy graphs are selected with this figure
Current	Transient current used in the cladding process
Voltage	Transient voltage used in the cladding process
Robot TCP Speed	Transient robot TCP (Tool Center Point) velocity
Weaving frequency	Frequency for weaving pattern used in the cladding process
Torch angle	Angle for the tool tip, between -90 and 90 degrees
Arc Lenght correction	Regulates the heat energy and the size of the droplets (from -30 to +30)
wire feed (mm/s)	Feeding velocity of the wire in mm/s
wire feed (percentage)	Feeding velocity of the compared for maximum of 100%
Dynamic Correction	Adjusts the properties of short circuit break (from -5 to +5)
Wire feeder roller current	Current for the wire feeding roller, decline of the current is an indication of wearied rollers

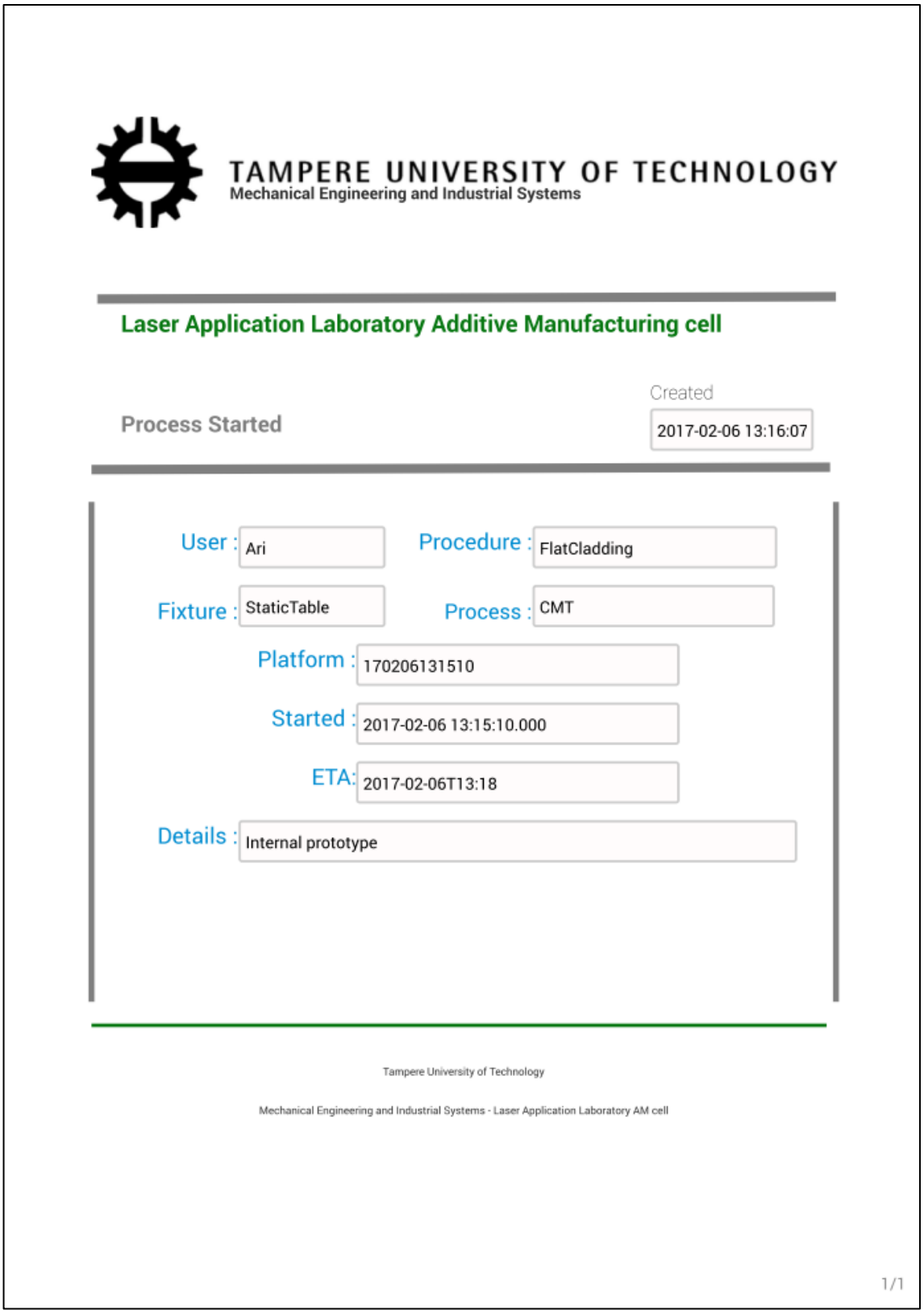
Gathered data for COAXwire process

Record name	Record definition
Timestamp	Timestamp from the start of the process
Laser ON/OFF	Laser Beam ON/OFF indication
Laser power	Transient laser power
Wire feed (mm/s)	Feeding velocity of the wire in mm/s
Wire feed started	Wire feed is active
Wire feed stopped	Wire feed is inactive
Robot TCP Speed	Transient robot TCP (Tool Center Point) velocity
Weaving frequency	Frequency for weaving pattern used in the cladding process
Torch angle	Angle for the tool tip, between 0 to 180 degrees

APPENDIX D: PROCESS DATA FILE EXAMPLE

```
170214120148,CMT,RodCladding,Internal,Ari,start,2017-02-14T12:01:48,0.01
0.001,1,0,0,2.31,1.15,0,0,0,50,0,0
0.011,1,0,0,2.31,1.15,0,0,0,50,0,0
0.022,1,0,0,6.97,3.49,0,0,0,50,0,0
0.032,1,0,0,6.97,3.49,0,0,0,50,0,0
0.043,1,0,0,6.97,3.49,0,0,0,50,0,0
0.053,1,0,0,11.68,5.84,0,0,0,50,0,0
0.064,1,0,0,11.68,5.84,0,0,0,50,0,0
0.074,1,0,0,16.39,8.2,0,0,0,50,0,0
0.084,1,0,0,16.39,8.2,0,0,0,50,0,0
0.095,1,0,0,21.1,10.55,0,0,0,50,0,0
0.105,1,0,0,21.1,10.55,0,0,0,50,0,0
0.115,1,0,0,21.1,10.55,0,0,0,50,0,0
0.125,1,0,0,25,12.5,0,0,0,50,0,0
0.135,1,0,0,25,12.5,0,0,0,50,0,0
0.145,1,0,0,25,12.5,0,0,0,50,0,0
0.156,1,0,0,25,12.5,0,0,0,50,0,0
0.166,1,0,0,25,12.5,0,0,0,50,0,0
0.177,1,0,0,25,12.5,0,0,0,50,0,0
0.187,1,0,0,25,12.5,0,0,0,50,0,0
      -------SLICED--------
12.424,1,0,0,25,12.5,0,0,0,50,0,0
12.435,1,0,0,24.52,12.26,0,0,0,50,0,0
12.445,1,0,0,24.52,12.26,0,0,0,50,0,0
12.455,1,0,0,21.87,10.94,0,0,0,50,0,0
12.465,1,0,0,21.87,10.94,0,0,0,50,0,0
12.476,1,0,0,17.38,8.69,0,0,0,50,0,0
12.486,1,0,0,17.38,8.69,0,0,0,50,0,0
12.497,1,0,0,17.38,8.69,0,0,0,50,0,0
12.507,1,0,0,13.48,6.74,0,0,0,50,0,0
12.518,1,0,0,13.48,6.74,0,0,0,50,0,0
12.528,1,0,0,10.08,5.04,0,0,0,50,0,0
12.539,1,0,0,10.08,5.04,0,0,0,50,0,0
12.549,1,0,0,6.99,3.5,0,0,0,50,0,0
12.559,1,0,0,6.99,3.5,0,0,0,50,0,0
12.57,1,0,0,6.99,3.5,0,0,0,50,0,0
12.58,1,0,0,4.35,2.18,0,0,0,50,0,0
12.59,1,0,0,4.35,2.18,0,0,0,50,0,0
12.6,1,0,0,2.26,1.13,0,0,0,50,0,0
12.611,1,0,0,2.26,1.13,0,0,0,50,0,0
12.621,1,0,0,0.87,0.44,0,0,0,50,0,0
12.632,1,0,0,0.87,0.44,0,0,0,50,0,0
12.642,1,0,0,0.87,0.44,0,0,0,50,0,0
12.652,1,0,0,0.28,0.14,0,0,0,50,0,0
end,2017-02-14T12:05:25
```

APPENDIX E: DESIGNED PROCESS REPORTS



TAMPERE UNIVERSITY OF TECHNOLOGY
Mechanical Engineering and Industrial Systems

Laser Application Laboratory Additive Manufacturing cell

Process Started

Created
2017-02-06 13:16:07

User : Ari
Procedure : FlatCladding
Fixture : StaticTable
Process : CMT
Platform : 170206131510
Started : 2017-02-06 13:15:10.000
ETA: 2017-02-06T13:18
Details : Internal prototype

Tampere University of Technology

Mechanical Engineering and Industrial Systems - Laser Application Laboratory AM cell

1/1

TAMPERE UNIVERSITY OF TECHNOLOGY
Mechanical Engineering and Industrial Systems

Laser Application Laboratory Additive Manufacturing cell

Process Report: CMT

Created
2017-03-05 12:39:58

Overall statistics of finished Additive Manufacturing process.

User : Ari

Procedure : GCode

Fixture : StaticTable

Process : CMT

Platform : 170305123251

Started : 2017-03-05 12:32:51.000

Ended : 2017-03-05 12:38:14.000

Details : internal prototype

Accessed CMT voltage and Current

Current average

Voltage average

Current Max

Current Min

Voltage Max

Voltage Min

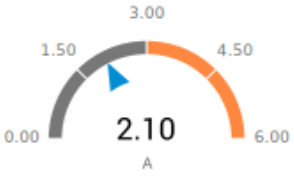
Tampere University of Technology

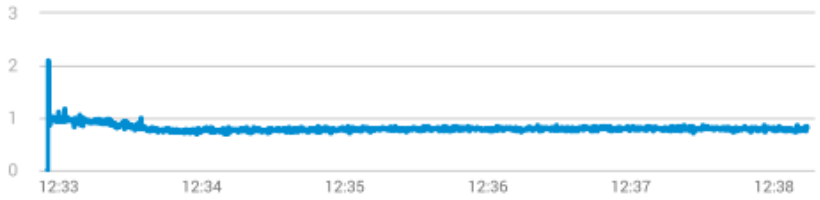
Mechanical Engineering and Industrial Systems - Laser Application Laboratory AM cell

Accessed CMT voltage and Current

Wirefeed roller current

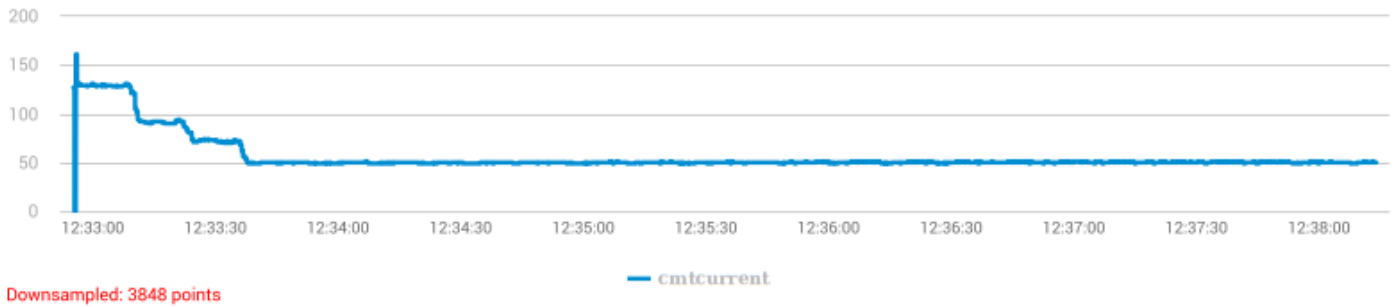
Higher than 3 amp roller current might be indication of wire trapping



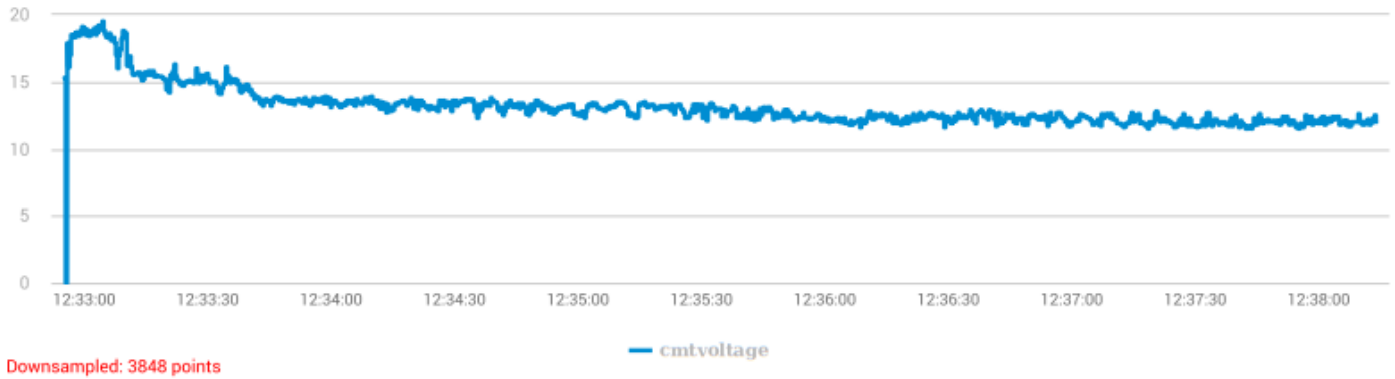


Downsampled: 2300 points

Voltage and Current



Downsampled: 3848 points



Downsampled: 3848 points

TAMPERE UNIVERSITY OF TECHNOLOGY
Mechanical Engineering and Industrial Systems

Laser Application Laboratory Additive Manufacturing cell

Process Report: COAXwire

Created
2017-02-06 14:20:59

Overall statistics of finished Additive Manufacturing process.

User : Ari

Procedure : FlatCladding

Fixture : StaticTable

Process : COAX

Platform : 170206141610

Started : 2017-02-06 14:16:10.000

Ended : 2017-02-06 14:18:00.000

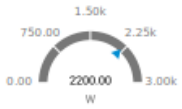
Details : Internal prototype

Accessed Laser power

Laser power average

Laser power Max

Laser power Min



Tampere University of Technology

Mechanical Engineering and Industrial Systems - Laser Application Laboratory AM cell