UNIVERSITY OF VAASA

SCHOOL OF TECHNOLOGY AND INNOVATIONS

AUTOMATION TECHNOLOGY

Vesa Iltanen

DATA COLLECTOR FOR INDUSTRIAL SANDING MACHINES

IoT Module Implementation

Master's thesis for the degree of Master of Science in Technology

Vaasa 27.5.2019

Supervisor	Jarmo Alander
Instructor	Thomas Höglund

PREFACE

This Master's thesis was conducted for Mirka Ltd department of power tools, Jepua, Finland.

I want to thank the supervisor of this thesis prof. Jarmo Alander and the instructor, MSc Thomas Höglund and give a big hand for the company of Mirka which made this possible.

I also want to thank my family for all the support and kicking my ass to graduate, also special thanks for all my friends that helped me thought the University.

Finally, thanks for all who had faith in me and it is great to end this journey with famous quote of Rick Sanchez: "Wubba lubba dub dub".

TABLE OF CONTENTS

SYMBOLS AND ACRONYMS

3GPP	the 3rd Generation Partnership Project
4G	Fourth Generation
5G	Fifth Generation
ADU	Application Data Unit
ANSI	American National Standards Institute
API	Application programming interface
ASIC	Application-specific integrated circuit
CRC	Cyclic Redundancy Check
DIY	Do It Yourself
EIA	Electronic Industries Alliance
EN	European standard
ETSI	The European Telecommunications Standards Institute
FPGA	Field Programmable Gate Array
GHz	Giga Hertz
GPIO	General-Purpose Input/Output
HDMI	High-Definition Multimedia Interface
HMI	Human–Machine Interface
IEC	The International Electrotechnical Commission
IEEE	Institute of Electrical and Electronics Engineers
IETF	Internet Engineering Task Force
IoT	Internet of Things
IP	the Internet Protocol
ISM	Industrial, Scientific and Medical

ISO	the International Organization for Standardization
L2CAP	The logical link control and adaptation protocol -layer
LPWA	Low Power Wide Area
LPWAN	the Low Power Area Network
M2M	Machine to machine
MAC	Media Access Control
Mbps	Megabits per second
MQTT	Message Queue Telemetry Transport
MTC	Machine Type Communications
mW	milliwatt
NAT	Network Address Translation
NB-IOT	Narrowband Internet of Things
NFC	Near-field communication
NTC	Negative temperature coefficient
OMA	Open Mobile Alliance
OS	Operating System
OSI	Open Systems Interconnection
P&G	Procter & Gamble
PCB	Printed Circuit Board
PDU	Protocol Data Unit
PLC	Programmable Logic Controller
PROFINET	PROcess FIeld NET
QoS	Quality of Service
RF	Radio Frequency
RFID	Radio-Frequency Identification

RPM	Revolutions Per Minute
RTU	Remote Terminal Unit
SCADA	Supervisory Control And Data Acquisition
SIG	Special Interest Group
SQL	Structured Query Language
SSH	Secure Shell
TCP	Transmission Control Protocol, and,
TIA	Telecommunications Industry Association
UHF	Ultra High Frequency
USB	Universal Serial Bus
VPN	Virtual Private Network
WAN	Wide area networks
WLAN	Wireless local area network
WPAN	Wireless Personal Area Network

VAASAN YLIOPISTO		
Teknillinen tiedekunta		
Tekijä:	**Vesa Iltanen**	
Diplomityön nimi:	**Data collector for industrial sanding machines**	
Valvojan nimi:	**Jarmo Alander**	
Ohjaajan nimi:	**Thomas Höglund**	
Tutkinto:	Diplomi-insinööri	
Oppiaine:	Automaatiotekniikka	
Opintojen aloitusvuosi:	2010	
Diplomityön valmistumisvuosi:	2019	**Sivumäärä: 80**

TIIVISTELMÄ

Tämän diplomityön tarkoituksena oli kehittää Mirka Oy:lle IoT-laite, joka kerää ja lähettää dataa palvelimelle analysointia varten. IoT-laite tulee sijoittumaan hiomakoneen yhteyteen, jossa sen tulee monitoroida laitteiden välistä kommunikaatiota. Tässä kommunikaation monitorointi tarkoittaa datan tarkastelemista ilman omien kutsujen lähettämistä laitteille, jolloin sen toiminta ei tule häiritsemään laitteita.

Hiomakone tuottaa jatkuvasti tietoa käytöstä ja laitteen olosuhteista piirikorttiin liitettyjen antureiden avulla. IoT-laite on tyypiltään datan kerääjä, joka tullaan liittämään Mirkan AIROS-hiomakoneeseen, jota ohjataan robotin välityksellä.

Työ on jaoteltu teoriaan, käytäntöön, analyysiin sekä tulevaisuuden suunnitelmiin. Työssä käydään läpi IoT:n arkkitehtuuria, laitteiden välistä kommunikaatiota sekä teknologioita, joiden avulla voidaan toteuttaa prototyypin suunnittelu. Lopuksi kerrotaan prototyypin suunnittelusta, testauksesta sekä analysoidaan saatua dataa. Viimeiseksi käydään läpi päätelmiä sekä parannusehdotuksia.

IoT-laite on toteutettu Raspberry Pi-korttikoneella, joka käyttää 4G-modeemia datan lähettämiseksi Mirkan palvelimen tietokantaan. Datan analyysissä on tutkittu laitteiden lähettämää käyttötietoa ja tietojen kuvantamiseen on käytetty Microsoft Exceliä. Ohjelman avulla muunnettiin tietokantaan kerättyä dataa visuaaliseen muotoon kuvaajien avulla, jotka hahmottavat saatuja tietoja.

Datan kerääjä on asennettuna Mirkan työpajalle, Jepualle, mutta tulevaisuudessa laite olisi tarkoitus antaa AIROS hiomakoneen ostajille osana toimituspakettia, asennetavaksi hiomakoneen yhteyteen.

AVAINSANAT: teollinen automaatio, esineiden internet, hiomalaite, tiedonkerääjä, prototyypin suunnittelu, robotiikka

UNIVERSITY OF VAASA
Faculty of technology

Author:	**Vesa Iltanen**
Topic of the Thesis:	**Data collector for industrial sanding machines**
Supervisor:	**Jarmo Alander**
Instructor:	**Thomas Höglund**
Degree:	Master of Science in Technology
Major of Subject:	Automation Technology
Year of Entering the University:	2010
Year of Completing the Thesis:	2019 **Pages: 80**

ABSTRACT

The purpose of this thesis was to develop an IoT device for a company called Mirka Ltd. The IoT device is designed for data collection from sanding machine and sending data to the server for later analysis. The IoT device will be located next to a sanding machine, where it will monitor communication between devices. In this case, the data collection is done without sending any data request to the sanding machine. This kind of data collection won't disturb the sanding machine's performance.

The sanding machine continuously produces data about the usage and conditions of the device. This data is received from built-in sensors which are located in the sanding machine's circuit board. The IoT device is a data logger that will be attached to the Mirka's AIROS sanding machine, which is controlled by a robot.

Thesis is divided into theoretical, practical, data analysis and future development parts. In the theoretical part the IoT architecture, communication between devices and technologies concerning the communication are discussed. After the theoretical part focus is on the IoT device itself, including the developed program, devices and tools that are used. The last part is about testing, data analysis and future development that are related to the data we collected from real use of the IoT device.

The IoT device is made using Raspberry Pi computer and 4G communication device. With the 4G device, IoT device is able to send data through the Internet to an external database, which is located in Mirka Ltd's server.

IoT device is set up and currently running in the Mirka Ltd's workshop in Jepua. In the future, the intention is to include it to the AIROS end product.

KEYWORDS: Industrial automation, Internet of Things, sander, data logger, protype development, robotics

1 INTRODUCTION

This thesis is a project for the company Mirka Ltd, it‘s power tools department which is responsible of the development and manufacturing of the company’s power tools including sanding and polishing machines**.** In this thesis we will be developing IoT device for monitoring data from sander machine and go through technologies behind the device and finally testing and analysis for prototype product. The name of the IoT device is AIROS logger which will be optional standalone component which will work with specific sanders. Because Mirka is operating globally we must ensure that our extractor is working fine despite of destination country or industrial communication environments.

1.1 Mirka

Mirka develops and produces sander products and it is part of the KWH Group, family-owned company. KWH group is divided into three independent divisions: KWH Invest, KWH Logistics and Mirka, which is the most important division of the company. (Mirka 2018a)

Mirka operates globally but it’s headquarter and production are located in Jeppo, Finland and subsidiaries are in Europe, Middle East, Asia, and North and South America. The company has high export rate, more than 97 % and products are sold over to 100 countries. (Mirka, 2018a)

The company is actively developing new products and improving the user experience, properties and quality of power tools. New products have to bring value to customers and one of the company’s main themes is to be innovative. One of the new products or rather applications is myMirka, which gives users more information on the machine’s usage and visualizes data from the sander. MyMirka was developed for data visualization and for analyzing the data of the sanders, thus the data analysis of this thesis is only a shallow analysis because Mirka Ltd already has an existing tool for that.

1.2 Target

One of Mirka's newest product innovation is automated random orbit sander, AIROS, this is made for robot use only. AIROS, that is illustrated in Figure 1, has been an investment for the future and it has a lot of new opportunities and potential to beat competitors and improve the company's services.

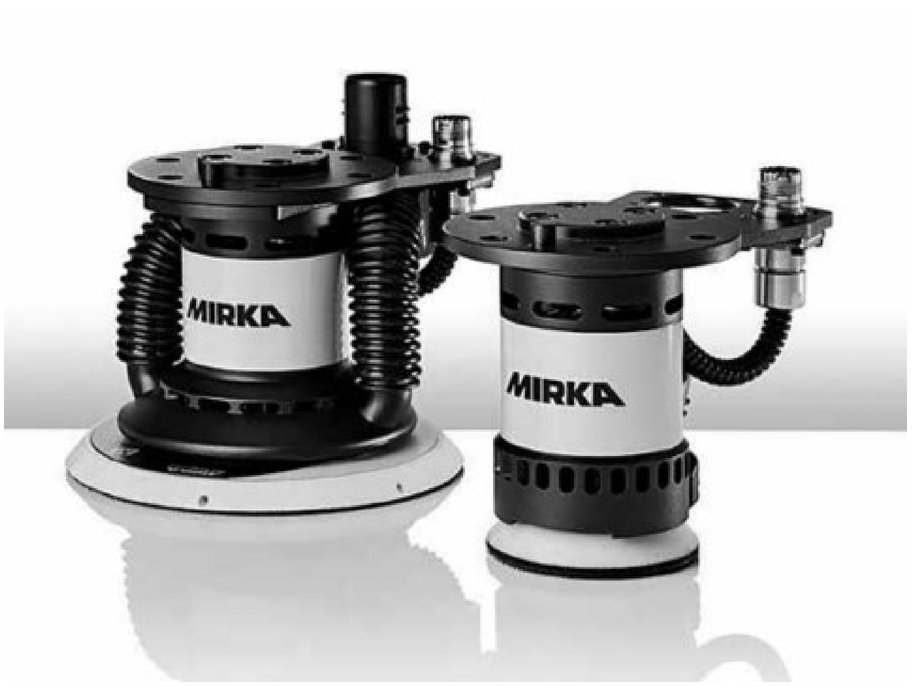


Figure 1. Mirka AIROS is available in two different sizes, in 150mm and 77mm (Mirka 2018b).

AIROS contains temperature sensors, one in tool and one in the motor drive. AIROS holds a lot of information on the usage and environment. There are for example information about how the voltage, current and speed changes and statistics like usage time and how the temperature behaves.

Now Mirka is interested to know how customers are using the products such as how long they run with it or is the temperature increasing during the work. Now all the interesting data isn't easily accessible and the customers and Mirka are more than eager to know more about it. So there is data, but we cannot access it easily without specific program.

Intention of the AIROS logger is to give it with purchaser of the AIROS. Mirka will ask permission to set it up and log the usage of the AIROS. The data is needed to improve AIROS and to detect the need of service. One possible future scenario for the data is to use it with the myMirka application which visualizes environment of the sanding machines like vibration and gives end-users access to it.

The AIROS logger, which will gather the data, must fulfill some major requirements: cellular communication, storing and sending data through VPN (virtual private network), reliability, controlling remotely and having to monitor the data packages.

In a nut shell, the thesis will investigate how to fetch and read raw sander data, way to send fetched data automatically to server for further usage and finally, how to improve existing and future products. Theory will descript ways to communicate with the AIROS and external server. In practice the AIROS logger will detect and send data.

1.3 IoT

Machine data has been collected a long time and computer driven machines can easily log desired data. Data can be sent wirelessly from long distances and capture and analyze signals locally or send it to cloud for later investigating.

The term Internet of Things was first coined by Kevin Ashton in his presentation conducted for Procter & Gamble (P&G) in 1999. Kevin Ashton was working to improve P&G business by linking its supply chain with Radio-Frequency Identification (RFID) information to the Internet. (Geng 2016)

The Internet of Things is creating intelligent connections between different diversities in the physical world and IoT tries to close the gap between the material and the information world. Commonly an IoT device is an embedded device or microprocessor that enables telecommunication. The IoT has two qualities: being an Internet application and dealing with information inside ‘things’. (Minoli 2013)

IEEE (Institute of Electrical and Electronics Engineers) has defined IoT as a system consisting of networks of sensors, actuators and smart objects and the purpose is to connect things to make them intelligent, programmable and more capable of interacting with humans and each other. (Geng 2016)

1.3.1 IoT products

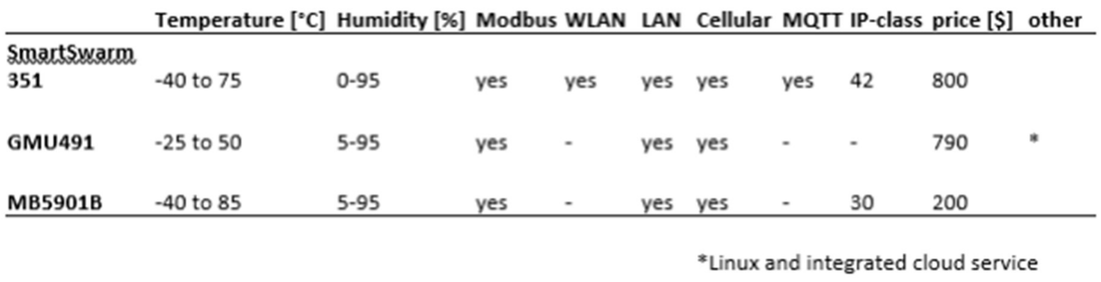
Internet of Things is a technology for digitizing the physical world and it is one of the most significant emerging technologies in recent years and it is said to be one of the prominent drivers of the fourth industrial revolution. It will have impact across business and industry around the world. (Geng 2016)

Devices that fulfill our requirements have already been developed or some devices can be extended with external components. Next, we will list a couple of commercial alternatives for a logging system that could be used with AIROS.

- SmartSwarm 351 from B+B SmartWorx company (Advantech B+B SmartWorx, 2019)
- GMU491 Cloud Gateway from Gluon (IonSign 2019)
- MB5901B Series from Atop Technologies (Atop 2019)

Chart 1. shows a small comparison for a couple of existing IoT devices or gateways that are suitable for industrial and logging use. These are useful for sending all and raw data to cloud for further investigation. However, there should be a program that detects useful data. Because all the data is sent to the server, fast connections and cloud performance is needed for working almost real-time.

Chart 1. Comparison of industrial IoT devices that partly fulfil demands of the logger.



	Temperature [°C]	Humidity [%]	Modbus	WLAN	LAN	Cellular	MQTT	IP-class	price [$]	other
SmartSwarm 351	-40 to 75	0-95	yes	yes	yes	yes	yes	42	800	
GMU491	-25 to 50	5-95	yes	-	yes	yes	-	-	790	*
MB5901B	-40 to 85	5-95	yes	-	yes	yes	-	30	200	

*Linux and integrated cloud service

The Chart 1. holds information about IoT devices which could be used for logging the communication. All devices can communicate with Modbus which is the main communication way of AIROS sander. They all also are able to use cellular communication and LAN (Local Area Network) and SmartSwarm is also able to communicate with MQTT. QMU491 runs with Linux so it could be used to monitor communication without sending own messages like the AIROS logger is planned to do. All these devices are expensive compared to self-made product which will cost approximately 100 euros or less and it is able to do all the same and more. The devices in the Chart 1. are industrial made devices and easy alternative for small scale logging.

These products are made for industrial use and they can handle multiple connections, and also serve as slaves or masters. The aim of the AIROS logger i.e. the target IoT device (Internet of Things), is to do almost same as the listed devices and more, but with less cost.

1.3.2 Security

IoT has grown fast and it has brought many types of issues and challenges. Security is one of the main issues for IoT technologies, applications and platforms. (Aldowah, Rehman, Umar 2018)

Probably the first security concern of the IoT is a compromise of the privacy and the protection of personal integrity. Massive growth of sensors in our environment, for example smartphones have lot of built-in sensors, huge amount of data can be collected from the people and it can be profiled and identified with analytics tools even if the data is anonymized. (Tsiatsis, Fikouras, Avesand, Karnouskos, Mulligan, Holler, Boyle 2014)

The Internet has made it possible to make economical or social damages but Internet connected and controllable devices have made it possible to do physical damage to people's property and also people's safety is at risk. (Tsiatsis i.e. 2014)

Security threats are classified as physical attacks, logical attacks and data attacks. Physical attacks are physical thefts, modifications to the software and side-channel attacks, which means attacks using information that is gathered from implementations of the computer system. (Amodu and Othman 2018.)

Logical attacks are impersonations, denial of service and relay attacks, meaning data capturing and replaying the data. The technique is common in keyless car systems where key data is captured and replayed. (Aldowah i.e. 2018)

Data attacks are privacy, data alteration and selective forwarding attacks, meaning dropping only selected data and forwarding other data normally. Devices and systems have to fulfill some security requirements, because of law and for safety. There are also security concerns like trust creation, advanced credential management for mobile communication and horizontal end-to-end security solutions. (Amodu i.e. 2018)

Amodu (2018) writes in the article that IoT security solutions should include effectively supporting authentication, proper identification of entities, confidentiality, data integrity, access control, and non-repudiation. Chart 2 presents some common security issues that can occur to any IoT devices but especially to the AIROS logger. The chart presents attack names in italics, they are most considerable attacks that this AIROS logger can front. AIROS logger uses VPN (virtual private network) connection to the Mirka Ltd server and the logger can be used for getting inside the company's network and information. The AIROS logger can be also captured and used for distributed denial-of-service attack for crashing servers. This hijacking is one of the biggest threats if the logger's software won't be updated regularly.

The first wave of the IoT attacks were in the 2016 when devices were turned into the botnets, known as Mirai Botnet. In the first half of 2018 were almost three times more samples detected than in the 2017 and almost ten times more than 2016. In the 2018, most of the attacks were trying of cracking the Telnet or SSH passwords and intentions is to get control of devices. After the access attacks it most likely downloaded one malware of the Mirai family. (Kuzin, Shmelev, Kuskov 2018)

Chart 2. The Security issues and probability of attacks of the IoT devices and most common attacks in the 2016 where trying to get control of the device.

Name	Description
Privacy and personal integrity	Data can be profiled and identified with analytics even it is anonymized
Physical attack	Physical theft, modifications to the software and side-channel attacks
Logical attack	Impersonation, denial of service, and relay attacks
Data attack	Privacy, data alteration and selective forwarding attacks
Control of device	Possible to do physical damage to property and also people’s safety are at risk

Most of the issues can be occurred when the attacker get inside the IoT system. The home automation systems can include security camera or microphone that can be used against owner. The AIROS logger won’t include confidential information about the company but hijacking to botnet or using VPN connection to unauthorized accessing to local network of Mirka have to be blocked.

2 IOT ARCHITECTURE

The Internet of Things is nowadays a popular concept, but it isn't a new idea to connect things together. Machine to machine communication (M2M) has been available for years before the IoT and they have lot in common but also a lot of differences, thus people usually confuse the terms. The focus of this chapter is to understand the structure and concept of the IoT and how it differs from an older M2M concept.

So, what is IoT? IoT is a vision of extending Internet connectivity to the things and things will be able to interact with real-world objects, process information and allow them to make independent decisions to help and save our time. IoT is nowadays on everyone's lips and it is said to be next step for the Internet and it will attach Internet to everything around us. IoT device is created to work with standard IP-protocol, so it doesn‘t need new network environments and the IoT device itself isn't so interested about the network under it. (Swetina, Lu, Jacobs, Ennesser, Song 2014)

M2M is defined as technologies that connect devices and machines to the Internet and transforms them into intelligent assets (Rawat, Singh, Bonnin 2013). M2M devices communicate between the same type of devices and specific applications using wireless or wired communication networks (Tsiatsis i.e. 2014). M2M solutions allow end-users to observe and process the data from sensors and events (Rawat i.e. 2013). M2M devices are mostly connected to the Internet via public telecommunication network and specific mobile networks. M2M device is very interested of the network beneath it because networks are not designed to device like, because M2M device uses very small amount of data and networks are designed for large data transfers and high speed (Swetina i.e. 2014).

What is the difference between these two technologies? IoT and M2M both connect sensors or other devices to information and communication systems, but the size of the system is different. IoT is the whole system with group of technologies, systems and design mechanics. IoT system can be very similar to the Internet of today, where things or devices can be interact with each other. M2M acts like enabler, inform changes but they won’t be as interactive as IoT device and usually has no or very simple interface like soda machines have. In this thesis it is not relative to go deeper in differences between concepts. (Haidine, El Hassani, Aqqal, El Hannani 2016)

2.1 Standardization

IoT system is based on many standardize bodies and it is important to recognize why they are developed and why it is important to follow them.

Standardization has been quite challenging because of the wide range of M2M applications and several standards development organizations are making efforts to facilitate M2M communication. These are 3GPP (the 3rd Generation Partnership Project), ETSI (The European Telecommunications Standards Institute), IETF (Internet Engineering Task Force), oneM2M, and NB-IOT (Narrowband Internet of Things). 3GPP provides a stable environment for members to produce reports and specifications that are used to define its technologies like radio access, core transport network and service capabilities. ETSI develops standards that are globally applicable for information communication technologies such as fixed, mobile, broadcast, radio and Internet technologies. IETF is developing towards operation of the Internet and the evolution of its architecture. OneM2M is targeted for M2M and IoT and develops required specifications. It tries to create a standard uniform service layer that can be embedded in various hardware and software. NB-IoT is a narrowband radio technology standardized by 3GPP, aimed at addressing requirements of IoT. (Amodu i.e. 2018: 265)

Due to continuous growth of the M2M field many standardization bodies were forced to define M2M standards and it lead to overlapping of different standards. After the overlapping mess, OneM2M was formed to create a global M2M specification and standards. (Ptiček, Čačković, Pavelić, Kušek, Ježić 2015).

Standardization of the IoT has concerns with representation formats, data dissemination mechanisms and data management platforms. Management of data from different sources must be flexible and help ecosystem services and innovations. Flexible data management will prevent ecosystem isolation but at the same time it also enables issues at security and information quality like trustworthiness and reliability.

2.2 Architecture

Next the architecture of the oneM2M is described. The oneM2M layered model is supporting end-to-end M2M services and it consist of three different layers: Application, common service and network service layers, see Figure 2. (oneM2M 2018: 535)

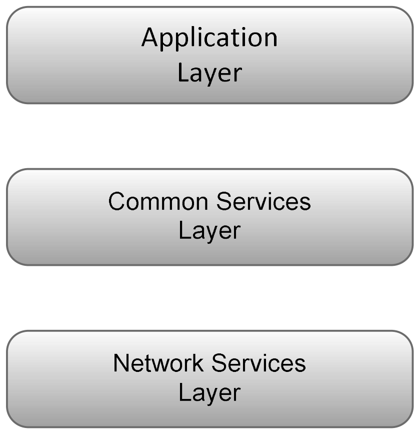


Figure 2. OneM2M layered model (oneM2M 2018: 535).

OneM2M is usually descripted to be application layer of IoT model and top three levels of OSI (Open Systems Interconnection) model: application, presentation and session layers. Other four levels of the OSI model are transport, network, data link and physical layer.

M2M and IoT manufacturers can use either a vertical or horizontal service model which the oneM2M architecture uses, see Figure 3. In the *Vertical* business model, one company is providing and controlling the IoT or M2M device, the gateway and the cloud-based service so the whole system is depending on one actor (Quinnell 2013). The vertical model uses point to point communication and it can be used at home, in automotive and industrial domains (Carey 2017). It has advantages because end-users don't have to worry about device compatibly and it has only a single point to contact if something goes wrong. Disadvantages are that the end-user is locked to a vendor who can decide on improvements, enhancements or upgrades.

Figure 3. also illustrates *Horizontal* architecture, which based on common layer and applications share common service layer, network infrastructure and it also uses multipoint communication. Applications share common infrastructure, environments and network elements (Carey 2017). Using a horizontal model allows multiple providers to use a common framework. Then it can be assumed that gateway and cloud resources are in place and have known and open functionality. It is easier to share data and resources between different devices and services. (Quinnell 2013)

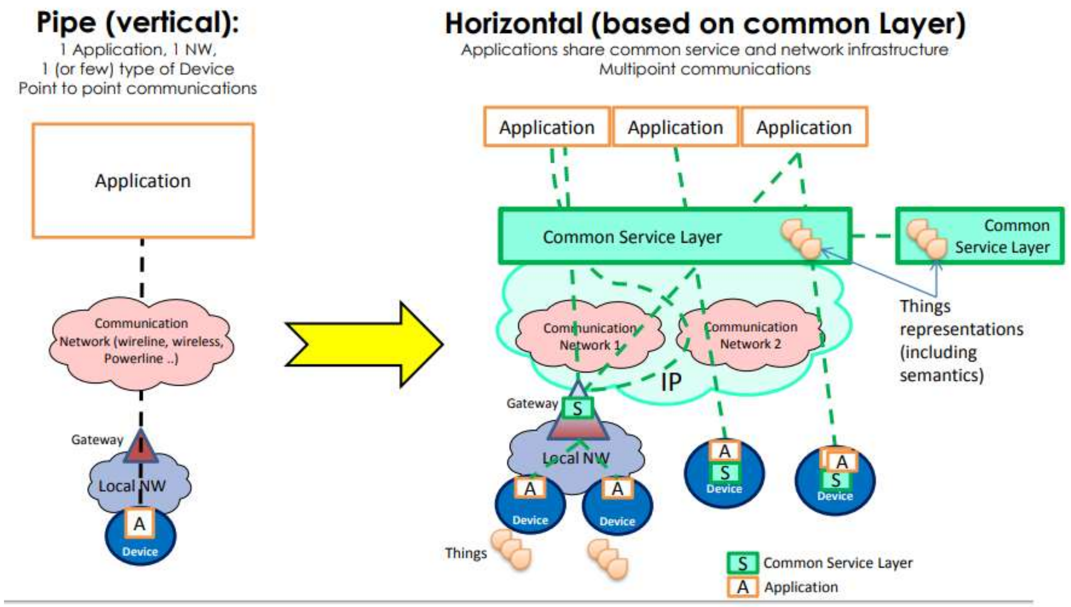

Pipe (vertical):
1 Application, 1 NW,
1 (or few) type of Device
Point to point communications
Application
Communication Network (wireline, wireless, Powerline ..)
Gateway
Local NW
A
Device
Horizontal (based on common Layer)
Applications share common service and network infrastructure
Multipoint communications
Application
Application
Application
Common Service Layer
Common Service Layer
Things representations (including semantics)
Communication Network 1
Communication Network 2
IP
Gateway
S
Local NW
A
Device
Things
S Common Service Layer
A Application

Figure 3. OneM2M horizontal architecture (multipoint communication) is evolved from the vertical, point-to-point communication architecture (Carey 2017).

Figure 3. illustrates how the oneM2M architecture is evolved from the traditional, closed pipe architecture to the horizontal approach. Horizontal shows that devices (dark blue circles), can use either own common service layer (green) or gateway (triangle) of the local network can handle the service. Because devices share common infrastructure, environments and network elements they can be used from multiple applications instead of specific application like in the pipe approach (connections: green dashed line). Devices are using Internet Protocol for the communication (pink cloud) instead of for example power line communication.

OneM2M functional architecture can be divided in three entities: application, common service and network service. Application entity represents application logic for end-to-end M2M solutions. Common service entity represents the set of common service functions of the M2M environments and network service entity provides services from underlying network to common service entities i.e. provide data transport services between entities like device management or device triggering. (Ptiček i.e. 2015)

oneM2M uses a resource-based data model and all oneM2M services are represented as resources. A resource is a data structure that can be uniquely addressed by a uniform resource identifier and oneM2M uses a tree-based resources structure. oneM2M has also adopted some of the standardized communication protocols such as MQTT (Message Queue Telemetry Transport), more on this in the communication layers and technologies section 3. (Andrianto, lam, Widodo, Lee, Lee, Lim 2018)

The IoT architecture can be divided in three different layers: application, communication and physical layer. The application layer is the layer for users where they can control the device and data. The communication layer is where all data communication is made, local and external communication. The physical layer has sensors and controllers and their interaction with a gateway. (El-Shweky, El-Kholy, Abdelghany, Salah, Wael, Alsherbini, Ismail, Salah, AbdelSalam 2018)

2.2.1 IoT application layer

The IoT application layer is responsible for providing services and defines a set of messages. An IoT device always has some sort of data processing environment for fetched data and making the data usable. It can be done directly by the application or globally by clouds, which can for example analyze, sort and store the data and use website as interface.

The application layer also provides device management and a virtual service layer that is responsible for data transport, security and service discovery. A virtual service layer provides information collected from objects and the performance of the actuators. (El-Shweky i.e. 2018)

2.2.2 IoT communication layer

The communication layer is the main channel between the application layer and different operating activities in the IoT system. It provides communication protocols for connected nodes because the data needs to be shared between nodes. Communication can be divided into three categories according to distance, short like Wi-Fi, medium like Ethernet and long-distance communication like cellular communication. (El-Shweky i.e. 2018)

2.2.3 IoT physical layer

The layer mostly consists of sensors that bring information for the system and actuators that do actions response to instructions from the system. They work together like temperature sensor tells heating must turn on. Sensors are devices that measures physical quantity and send feedback to the controller which send signal to the actuators to do actions for example to turn on heating. (El-Shweky i.e. 2018)

2.3 Summary for AIROS logger system

Planning of the new IoT-system needs different aspects as mentioned in the previous subchapters focusing on standards and architecture. There are multiple ways to achieve fully functional IoT-system but making it more flexible and better it needs to follow standards and planned architecture.

The chapter two also gave understanding of the concept and architecture of the IoT. This knowledge will be needed when the architecture of the AIROS logger is planned. There are many ways to create IoT system, so the architecture of the AIROS logger needs to be defined. Decision between the vertical and the horizontal architecture depends what kind of system is planned to build.

This decision also has effects to numbers of the standards that the AIROS logger have to fulfil. If prototype device needs to be able to communicate with different IoT systems or different vendors devices, the IoT architecture needs to be horizontal. Then it needs to fulfil some IoT standards to able communicate smoothly without specific integration. If prototype is planned to be closed system, IoT architecture can be vertical. Using the that approach, IoT standards can be partly ignored and use own technical choices.

The system of the AIROS logger is planned to be simple to develop, so the selected architecture approach is selected to be vertical. It is selected because the wanted system is closed and it is not for commercial use or planned to be integrated to other IoT system. The AIROS logger system will be using only commonly used standards that are known for example from phones and Internet. The system won't use any specific IoT or M2M standards.

Layers of the IoT architectures: physical, communication and application are worth defining before starting to develop the device. The physical layer of the logger is responsible for, for example converting digital binary to electrical messages.

The communication layer of the AIROS logger provides protocols for the communication that physical layer enables. Modbus and HTTP protocols are provided from the communication layer. The Modbus is used between the AIROS and robot, and HTTP is used between the AIROS logger and the server, technologies descripted in Figure 4 .

The application layer is the top layer and it is responsible for using the data that physical and communication layers provide. The application layer includes the monitoring application that detects data that transports between the AIROS and robot. The application also needs to convert, save and check monitored values. Figure 4. illustrates diagram of the AIROS logger that shows, what kind of communication the system needs, Chapter three gives more details on the communication methods of the AIROS logger.

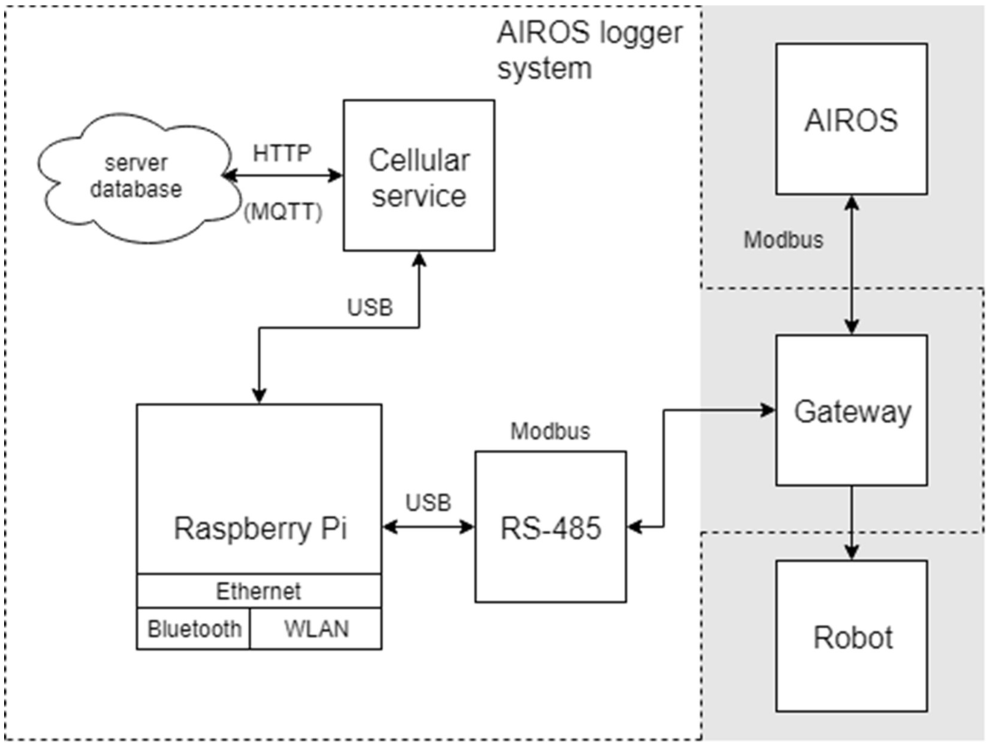


Figure 4. AIROS logger diagram with communication.

3 SELECTING COMMUNICATION LAYERS

Internal communication devices, logger and the server, must naturally communicate with each other and communication can be established in different ways. Communication is often classified by distance and the scale starts from hundreds of nanometers and ends at interplanetary size, so there is a wide range of communication types which can be used. Familiar communication protocols are NFC (Near-field communication), Bluetooth, Ethernet, Wireless local area network (WLAN) and wide area networks (WAN) like fourth generation (4G) mobile network. The core idea of the IoT is to use the IP (Internet Protocol) and mainly IP version 6 (IPv6) to communicate between device and cloud services. Today there are several standards available for the IoT on different levels of the protocol stack and they can be combined in different ways. (Larmo, Ratilainen, Saarinen 2018)

Different communication technologies are used in different situations. Next are listed and explained the most used and common communication protocols and technologies for the IoT.

First part contains technologies that are exist in current AIROS system: Modbus, RS-485 and Ethernet. Second part contains technologies that will be implemented to the AIROS logger, these are Bluetooth, WLAN, Cellular services and HTTP. Third par is more about IoT technologies like should consider in the future use, MQTT, and also technologies that designers should know, CoAP and Lightweight M2M. Final subchapter describes what and where technologies are used or have potential later use in AIROS logger.

3.1 AIROS Technologies

AIROS uses wired technologies which are oldest and the most reliable way to communicate even if the technologies have wireless alternatives. Next is descripted couple communication protocols and standards that AIROS logger might use or AIROS sanding machine is using.

3.1.1 Modbus

The AIROS supports the Modbus communication protocol and it is mainly way to communicate with the device. The AIROS logger needs this protocol in some stage because communication is handled between the AIROS and robot with this protocol. Modbus protocol is needed to implement to the application of the AIROS logger.

Modbus is an application layer messaging protocol that provides client-service communication between devices. The Modbus protocol implements client-service architecture and operates basically in request-response mode. The Modbus client-service model consist of four types of messages. 1) Request, client sent message that initiates transaction. 2) Confirmation, client-side response that indicates that the message has been received. 3) Indication, server-side message that request is received. 4) Response, server sent response message. (Reynders i.e. 2004)

The client-server model is used to exchange real-time information between device applications and devices, two device applications or devices and HMI/SCADA (SCADA, Supervisory Control And Data Acquisition and HMI, Human-Machine Interface) applications. (Reynders i.e. 2004)

Information exchange flow between client and server starts with the client, master device, initiates a request. It generates a protocol data unit, PDU, consisting of function code and data request. The PDU is converted to an application data unit, ADU, with added bus or network related field such as slave address and checksum for error detection. Modbus frame with ADU and PDU is illustrated in Figure 5. (Reynders i.e. 2004)

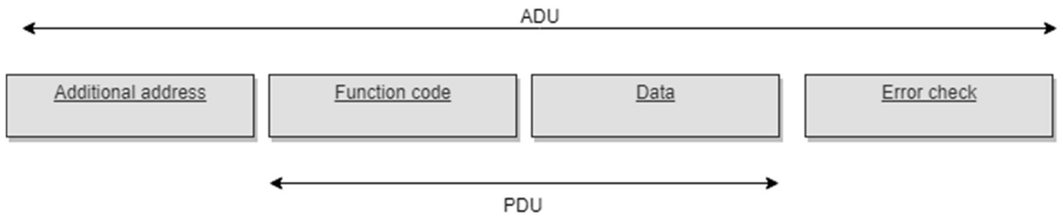


Figure 5. Generic MODBUS frame.

After the server has performed required action it initiates a response. The full interaction between client and server is illustrated in Figure 6.

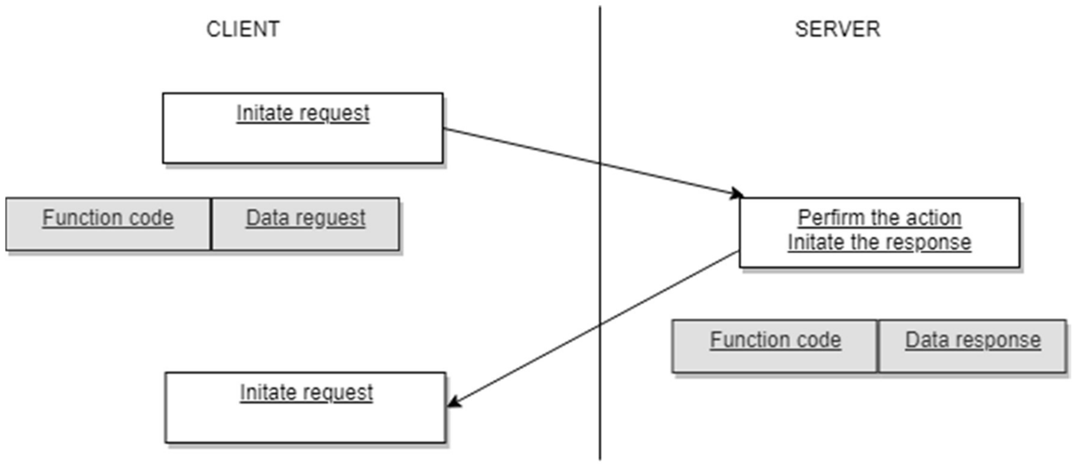


Figure 6. MODBUS transaction.

Modbus needs additional support from the lower layers to get the messages across. A popular method is to use master-slave layer 2 protocol, transmitting the data in serial format over for example RS-232 or RS-485 which is also known as ANSI/TIA/EIA-485 (ANSI, American National Standards Institute; TIA, Telecommunications Industry Association; EIA, Electronic Industries Alliance). A newer approach would be to use TCP/IP and Ethernet to convey data from client to server, which requires additional sublayer to map Modbus application layer on to TCP. Sublayer encapsulates the Modbus PDU, so TCP/IP can be sent as a packet of data, Figure 7. illustrates the communication. (Reynders i.e. 2004)

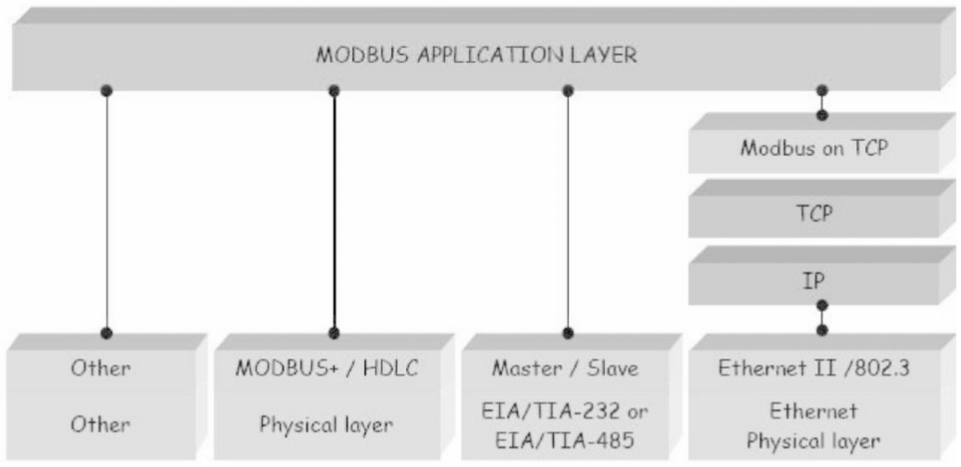


Figure 7. Modbus communication stack (Reynders i.e. 2004).

Modbus message frame is illustrated in Figure 5 and the first field of this frame is the address field, which every message contains and it consists of a single byte of information. In request, byte identifies the controller to which the request is being directed and in response, frame begins with the address of the responding device. The maximum number of slaves can be between 1 to 247 which is also address numbers limitation. Typically, there is one master and a few slaves. (Reynders i.e. 2004)

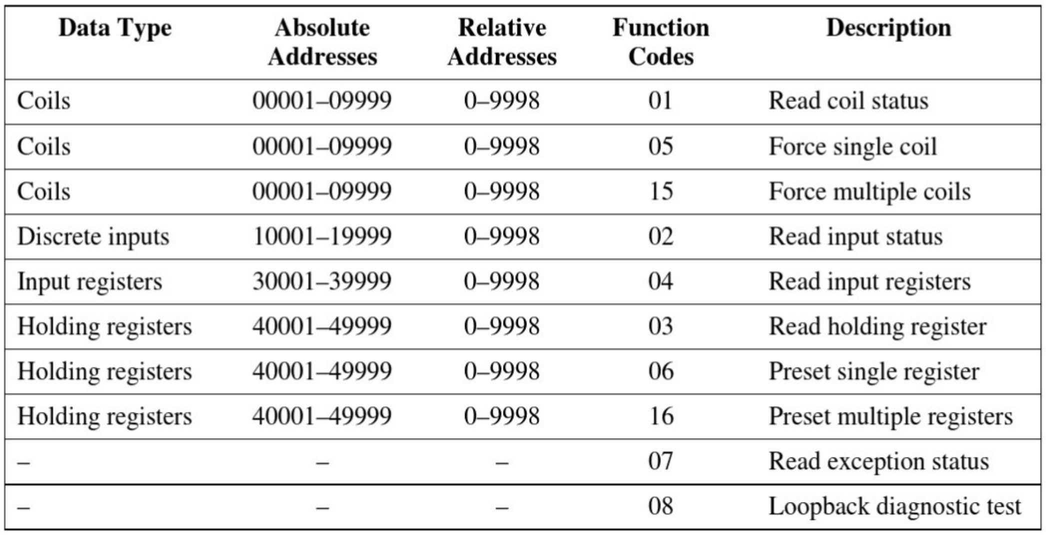
The function field is the second field of the message frame, which also consist of a single byte of information. It identifies function that target device like PLC (Programmable Logic Controller) is to perform. If target device can perform the function, the function field of its response will echo that of the original request. Otherwise, the function field of the request will be echoed with its most significant bit set to one, signifying an exception. (Reynders i.e. 2004)

The message field is the third field of the message frame, the length of which varies according to which function is requested. In the request, the message may contain information that is needed to fulfill the requested function and in the response, the message contains data that is requested by the host. (Reynders i.e. 2004)

The error-check field is the last message frame and it consists of two bytes. The message frame is calculating the value of this field by performing a cyclic redundancy check (CRC-16). This is done to check for errors if the message is damaged during the transmission and preventing devices from reacting. (Reynders i.e. 2004)

There are lot of Modbus function codes and some typical function codes are illustrated in **Error! Reference source not found.** which also shows typically data types and their descriptions. The requested data will depend on what function code is used and typically used function codes are from 01 to 07. Next is explained more about the function code 01 and function code 02. (Reynders i.e. 2004)

Chart 3. Examples implementation of Modbus addresses and function codes (Reynders i.e. 2004).



Data Type	Absolute Addresses	Relative Addresses	Function Codes	Description
Coils	00001–09999	0–9998	01	Read coil status
Coils	00001–09999	0–9998	05	Force single coil
Coils	00001–09999	0–9998	15	Force multiple coils
Discrete inputs	10001–19999	0–9998	02	Read input status
Input registers	30001–39999	0–9998	04	Read input registers
Holding registers	40001–49999	0–9998	03	Read holding register
Holding registers	40001–49999	0–9998	06	Preset single register
Holding registers	40001–49999	0–9998	16	Preset multiple registers
–	–	–	07	Read exception status
–	–	–	08	Loopback diagnostic test

Function code 01 means reading one or multiple coils of the target device, the response represents on/off status of the logic coils. The data field contains information about how many coils will be read and the response contains that many bytes of coil data. (Reynders i.e. 2004)

Function code 02 is reading digital input status and it enables the host to read one or more discrete inputs in the target device. The request frames data field consist of the relative address of the first discrete input and after it follows information on how many discrete inputs will be read. The response data consists of count of discrete input data bytes and after it bytes of discrete input data. (Reynders i.e. 2004)

3.1.2 RS-485

RS-485 is usually used to implemented to Modbus which uses it as physical layer. RS-485 refers to Regulation specification and the RS-485 is one of the used RS interface standards. The standard is an extension of the RS-422 but it increases the number of transmitters and receivers permitted on the line. RS-485 and RS-422 have the same distance and data speeds. It allows a ‘multidrop’ network connection on two wires. RS-485 allows communications at distances of up to 1200 meters and data rates up to 10 Mbps. It can have up to 32 line drivers and up to 32 receivers on the same line. It doesn’t achieve maximum bit rate and maximum length at the same time. (Reynders i.e. 2004)

Conductor markings A in B are mentioned in the RS-485 specification. Conductor A is known as A-, TxA and Tx+. Conductor B is similar and known as B+, TxB and Tx-. Identifying cables can be done as follows: in the OFF-state, e.g. -8V. The voltage of the A wire is more negative than on the B wire. The differential voltages on outputs A and B of the driver are in the range -1.5 to -6 V on terminal A with respect to the terminal B for a binary state 1, and in the same range but positive values for binary state 0. (Reynders i.e.2004)

3.1.3 Ethernet

Ethernet is widely used for computer networking and it is hard to find a device that isn't using Ethernet in some level. Ethernet was first introduced in 1980 and it got internationally standardized in 1983 as IEEE 802.3. Network speeds range from 10 Megabits per second to 100 Gigabits per second, depending on which cables are used. Physically it uses coaxial cables, twisted pairs or optical fibers. Ethernet is usually used in local area networks, metropolitan area networks or wide area networks. Ethernet is included in the datalink layer of the OSI model. (El-Shweky i.e. 2018)

Because Ethernet supports high speed communication, it is ideal for applications that handle a huge amount of data and require high speed. Ethernet cables are ideal for transmitting data to remote destinations, but it also has disadvantages compared to other wireless communication protocols. It needs a direct physical connection between nodes and it is vulnerable to physical damage. (El-Shweky i.e. 2018)

3.2 Additional technologies for the AIROS logger

This Chapter 3.2 gives more details about technologies that are implemented to the AIROS logger system. Modern solutions usually uses wireless alternatives to connect to IoT devices and these technologies that are descripted in this section is more about wireless solutions. Selecting most suitable wireless technologies from huge scale of technologies may be hard. Figure 8 tires to illustrate the suitability of different technologies to different IoT environments.

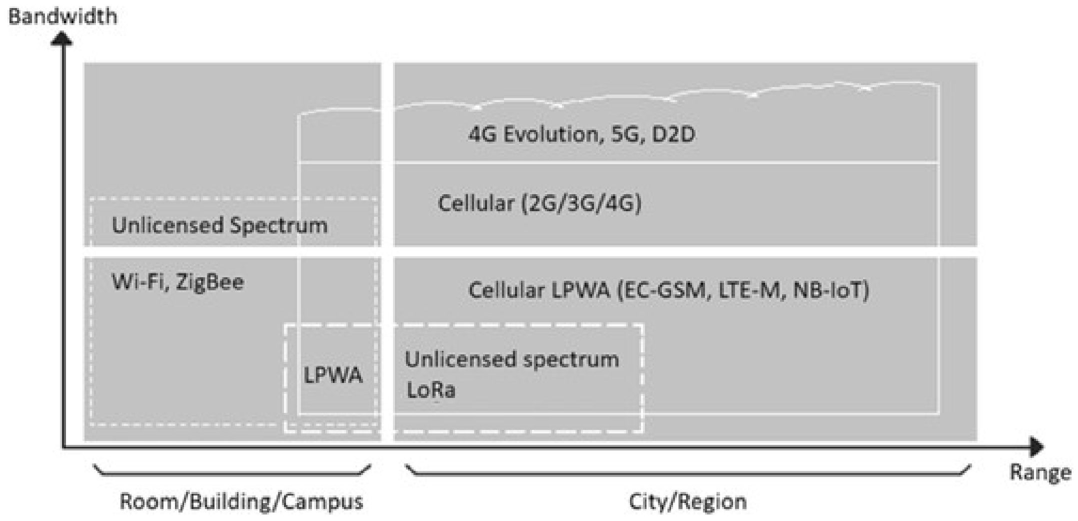


Figure 8. Suitable wireless technologies for different IoT environments (Sohraby, Minoli, Occhiogrosso,Wang 2018).

There are roughly five different approaches to wireless radio technologies: cellular, LPWA (Low Power Wide Area), satellite, short range wireless, wireline to wireless like Wi-Fi.

3.2.1 Bluetooth

The AIROS sander has the option to communicate with Bluetooth but communication can be low-quality or even impossible. Quality depends on how far the AIROS logger is from the AIROS and if there is some material between them. That is why this option is hardly considered now but in the future it could be possible.

Probably the most familiar short-range wireless technology is Bluetooth, which is a WPAN (Wireless Personal Area Network) technology that uses radio waves instead of wires or cables to connect devices. Bluetooth is one of the most popular technologies in consumer electronics and it is based on the IEEE 802.15.1 standard. A Bluetooth device contains a computer chip with a Bluetooth radio and software. “Communication between Bluetooth devices happens over short-range, ad hoc networks known as piconets. A piconet is a network of devices connected using Bluetooth technology.” Bluetooth is a master-slave connection and uses usually 2.4 – 2.48 GHz UHF (Ultra High Frequency) radio waves or other ISM (Industrial, Scientific and Medical) bands. Bluetooth uses frequency hopping spread spectrum and a full-duplex signal. The Bluetooth version 2.0 bandwidth is usually 1-3 Mbps, but newer versions can reach higher speeds. The system uses omnidirectional radio waves that can transmit through non-metal barriers like walls. Bluetooth was invented in the late 1994 and was considered a wireless alternative to RS-232 data cables. (Minoli 2013:170-177)

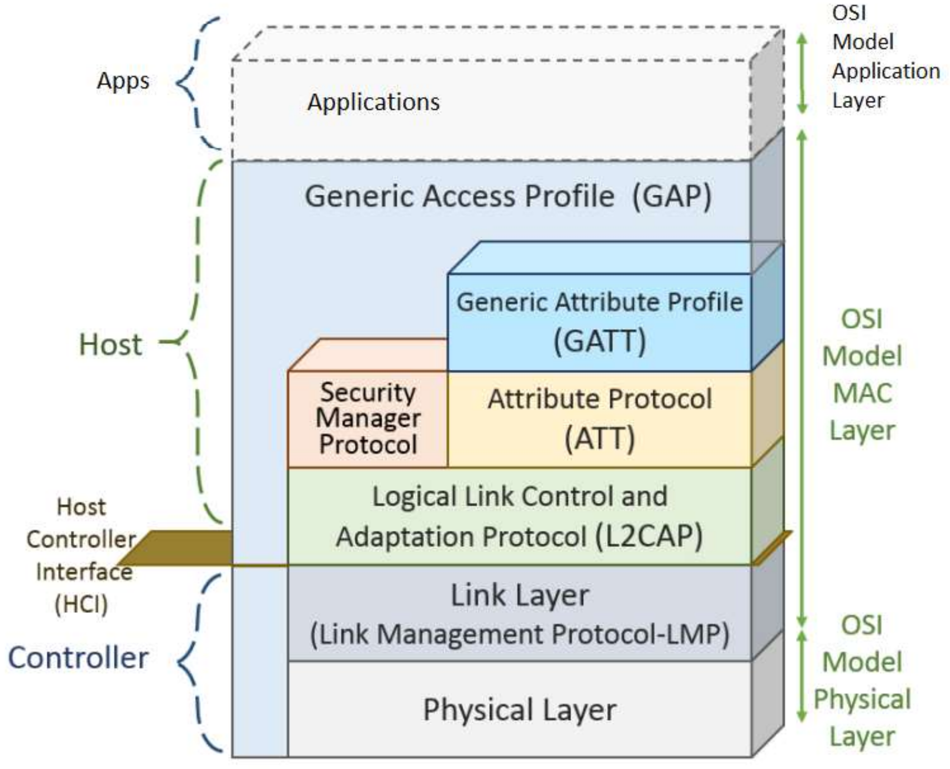

Apps
Applications
OSI Model Application Layer
Host
Generic Access Profile (GAP)
Generic Attribute Profile (GATT)
Security Manager Protocol
Attribute Protocol (ATT)
OSI Model MAC Layer
Host Controller Interface (HCI)
Logical Link Control and Adaptation Protocol (L2CAP)
Controller
Link Layer (Link Management Protocol-LMP)
Physical Layer
OSI Model Physical Layer

Figure 9. Bluetooth Low Energy core specification (Oliveira, Rodrigues, Kozlov Rabêlo, Albuquerque 2019: 9).

Bluetooth protocol layers are described in Figure 9 that shows how Bluetooth is being used. Layers of the Bluetooth stack does not fully match to the OSI model and they have some overlapping but next the Bluetooth Low Energy (BLE) protocol stack is roughly compared to OSI model. The BLE consist of controller, host and app part. The controller part is consists of physical and MAC (Multimedia Access Controll) layer, that is part of the link layer. The whole host part belongs to the MAC layer. The application layer is roughly same as application layer. (Oliveira etc. 2019: 9)

Bluetooth has evolved through four versions and all versions are downward compatible. In mid-2016 Bluetooth SIG (Special Interest Group) announced a new version, Bluetooth 5. It had been said to reach two times higher speed and quadruple the distance compared with the older 4th version of Bluetooth. It consists of three different power classes and the higher the class, the shorter the range and the lower the power class of the Bluetooth. The first class uses a power level of 100mW and it reaches up to a distance of 100 meters. The second class uses a power of 2.5 mW and reaches up to a distance of 10 meters, and the last class uses a power of 1 mW and reaches up to a distance of 1 meter. (Minoli 2013: 172)

Bluetooth standard IEEE 802.15 has four different sublayers: RF-layer (Radio Frequency), Baseband layer, the link manager, L2CAP (the logical link control and adaptation protocol), which is also a MAC-layer (Media Access Control) protocol.

“Overall Bluetooth is design for high QoS (Quality of Service) applications, a variety of duty cycles, and moderate data rates in networks with limited active nodes” (Minoli 2013:172)

Bluetooth allows devices to create a trusted relationship between a master and slaves. There can be up to seven slaves and this group of master and slaves are called piconets. The master can communicate only to one slave at a time but the master can change connection pair rapidly, which makes it seem like it communicates with multiple slaves at the same time; in a round-robin fashion. A Bluetooth device can act as a bridge between two piconets and at one time it can be the master in one piconet and a slave in another piconet. This is called a Scatternet when Bluetooth devices are connected to multiple piconets. (Minoli 2013: 178)

Hackers are already using Bluetooth to attack devices which means Bluetooth devices are becoming a security issue. A method called bluejacking is searching for available Bluetooth devices and sending unsolicited messages. Another type of attack is Bluesnarfing, which uses the same method to access contact lists and other information without the user's knowledge.

3.2.2 Wireless Local Area Network

Wireless Local Area Network usually refers to the Wi-Fi which is a trademark of Wi-Fi Alliance. Nowadays high-speed wireless area networks are very compatible with IoT and computer networks because of improved data rates and lower costs. First generation devices were only capable of speeds of 1 to 2 Mbps and nowadays it goes up to 7 Gbps (IEEE 802.11ad). (Olenewa, 2012)

3.2.3 Cellular service

Cellular networks like 3G, 4G or new 5G sound like practical solutions for IoT/M2M applications designed to operate in different places. It is often chosen because cellular networks can cover nationwide areas. In the near future, IoT and M2M applications are expected to be major contributors of traffic and also revenue for cellular networks. (Sohraby i.e. 2018)

It is important to choose the right cellular system from 3G, 4G or 5G or specific M2M cellular service for the IoT and M2M system. A smart grid environment is a good example of why it is so important to calculate which cellular to choose based on factors such as cost, reliability and deployment schedule and performance. IoT and M2M traffic has specific characteristics, including data priority, transaction size, the real-time streaming QoS requirements and higher delay tolerance of data. 3GPP is able to differentiate to Machine Type Communications (MTC) and supporting selectively MTC devices in case of network jam. Reliability and confidentiality are actual security concerns and endpoints need to provide and support virtual private network (VPN) constructs, embedded firewalls and other capabilities. (Sohraby i.e. 2018)

One aim with the AIROS logger is that it should be able to communicate independently without having to rely on local area networks. Cellular services are a good way to achieve this with low cost.

3.2.4 HTTP

HTTP (Hyper Text Transfer Protocol) is a standard protocol used on the Web and it is based on client-server communication. It is between client and server. A client is the Web browser or an application program that sets up a connection to send queries to the server. A server is an application that accept connection request and responds to queries. (Anttalainen 2003: 332)

An HTTP request identifies the web resource i.e. the URL that the client is interested in and what the server does with it. HTTP enables the fetching of different kinds of content from the server such as text, images and video. First, TCP connection is established for the HTTP to send requests and responses via this connection. The destination host is identified in the URL and DNS is also helping in this identification process. (Anttalainen 2003: 332)

HTTP request contains information on what the server should do with the given resource and HTTP response consists of a header which contains information about the resource and actual resource. Figure 10 illustrates the HTTP request and response procedure. (Anttalainen 2003: 332)

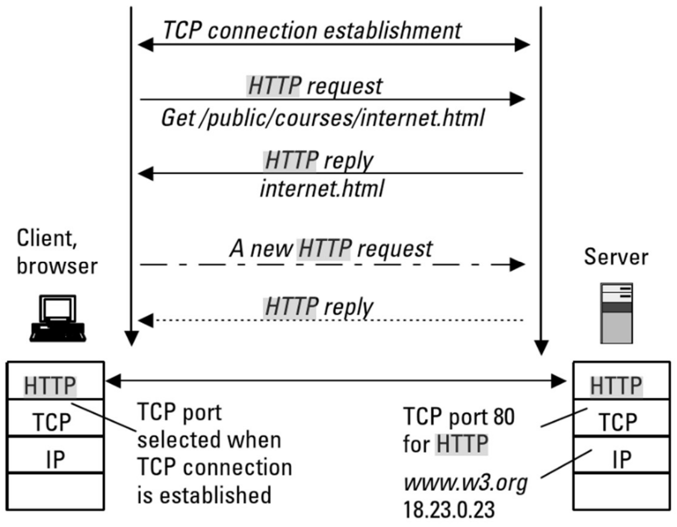


Figure 10. HTTP request and reply procedure (Anttalainen 2003: 333).

HTTP is not designed to devices used in IoT which have limited resources but it remains widely accepted within the existing network infrastructure and it is hard to find IoT system which isn't rely on HTTP in some degrees. There are some challenges when using HTTP in IoT: (Bziuk, Phung, Dizdarević, Jukan 2018: 350-355)

1) Latency: Server needs to receive and process a request from the client and this latency is a consequence of both HTTP and its underlying TCP. (Bziuk i.e. 2018: 350-355)

2) Energy and power consumption: Large uncompressed and redundant HTTP headers cause higher network usage and increased power consumption (Bziuk, i.e. 2018: 350-355)

3) No server push option: HTTP is request/response protocol, so a HTTP server has to wait for a specific request from the client before the data transfer is initialized. (Bziuk i.e. 2018: 350-355)

4) HOL (Head of line) blocking problem: the HTTP version 1 allows pipelining request multiple objects over the same TCP connection. Because of the request / response, server has to response request in order and an early request for a large

object as a lost request can delay all later pipelined requests. (Bziuk i.e. 2018: 350-355)

Even HTTP version 1 has issues. It is still an acceptable and widely used solution for IoT. Reasons why HTTP is still in use is that it is probably an easier and faster solution to implement than other protocols. (Bziuk, Phung, Dizdarević, Jukan 2018: 350-355)

3.3 Common IoT protocols

Next is briefly introduced some protocols that have not gained so much popularity at the moment in IoT or M2M systems. These aren‘t implemented anywhere in the AIROS logger IoT-system either but are well known and that’s why they are briefly introduced.

3.3.1 Message Queue Telemetry Transport

Message Queue Telemetry Transport (MQTT) is a connectivity protocol for IoT and M2M devices and it was invented in 1999 by Dr. Andy Stanford-Clark from IBM and Arlen Nipper from Arcom (Andrianto i.e. 2018). Most of the popular IoT applications use MQTT or CoAP as choice of the data transfers (Larmo i.e. 2018). MQTT is as an extremely lightweight publish/subscribe messaging transport and it has low power usage, small header and efficient distribution to one or many clients. (Andrianto i.e. 2018)

The principle of the publish/subscribe messaging is that there are entities that are interested in piece of information that will register their interest. The process of registering an interest is called subscription and the interested entity is called subscriber. Entities that want to produce information are called publishers. A broker, which is what is between a publisher and subscriber, ensures data delivery from the publisher to the subscribers. There are three different kinds of publish/subscribe systems: topic-, type- and content based. For example the topic-based system is where the list of topics is already known. A subscriber sends a message to the broker and the publisher sends a message which contains the data to be published. The broker will transfer the message to the subscriber if the topic of the publisher and subscriber matches. (Andrianto i.e. 2018)

MQTT also supports basic end-to-end Quality of service with three different levels, depending on the reliability of the message. Level zero is where a message is delivered once or not at all. In the first level, a message will be retransmitted until acknowledged by the receiver. Level two ensures reception of the message and it is delivered only once to the receiver. (Andrianto i.e. 2018)

3.3.2 Constrained Application Protocol

HTTP is very popular protocol but for tiny resource-constrained device that communicates over resource-constrained IP network, HTTPU is not practical option because it requires too many resources and too much bandwidth. Some networks limit the size of the datagrams and if a device is planned to be used in for example 6LoWPAN (IPv6 over Low power Wireless Personal Area Netroks), it will limit the size of the datagrams. Constrained Application Protocol (CoAP) is attempting to solve this problem (Waher 2015: 120-125).

CoAP designed by IETF and CoAP is a client-server or request and response protocol that fills the requirements of only constrained node with a low capability in RAM (Random-Access Memory) or CPU (Central Processing Unit) and constrained with lower power. CoAP supports publisher/subscribe architecture as previously mentioned MQTT. CoAP is a web transfer protocol and it holds up unicast and multicast requests by taking advantage of UDP. (Bahashwan, Abdullah, Manickam, Selvakumar)

3.3.3 Lightweight M2M

LWM2M is a device management protocol and it is designed for sensor networks and the demands of the M2M environment. Lightweight M2M is specifically designed for the management of low power and constrained IoT/M2M devices (Putera, and Lin 2015). LwM2M is developed by OMA specWorks, which is a successor brand for Open Mobile

Alliance and its purpose is to develop technical specifications in the mobile and IoT markets (OMA specWorks 2019).

Device management protocol is designed for reducing the amount of time to configure and manage the M2M devices, especially for those located in remote areas. This is done by enabling the device to collect resources and information as a managed object. Device management protocol has to be lightweight to fit the limited properties of IoT / M2M devices. Then it is possible to generate managed objects. (Putera and Lin 2015)

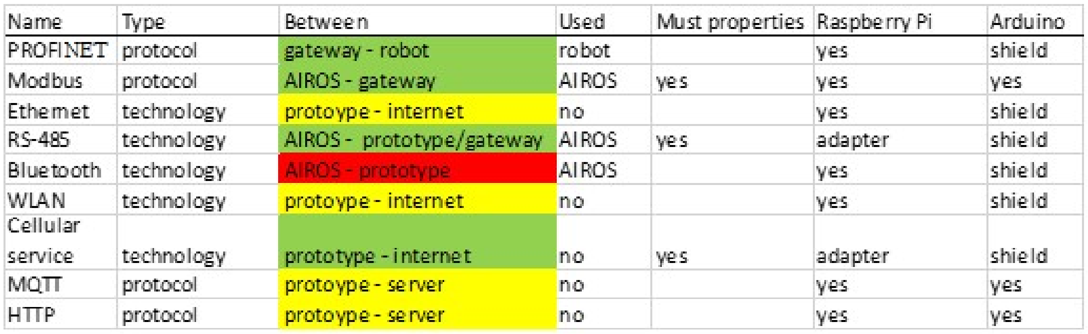
ETSI and OneM2M and other IoT / M2M standard organizations don't purpose any other device management mechanism. They have adopted the Device Management protocols by OMA for mobile devices and the TR-069 protocols by Broadband Forum for fixed devices. (Putera and Lin 2015)

3.4 Implementing to AIROS logger

The AIROS logger system can be divided in two parts: the system of AIROS sander which consist of communication between AIROS and other devices. See more details in chapter 4.3.2. The second part is the logger system which technology choices is described next.

Chart 4. shows a list of technologies and protocols that are described in this chapter and where it is used or needed for AIROS logger. The chart has a column that has colors that indicate green if it is used, yellow if it is wanted but not necessary and red if property is not needed. AIROS logger is supposed to work with cellular service so Ethernet and WLAN isn‘t necessarily needed but they give an opportunity to connect to the Internet if the cellular service fails. The possible server protocols are MQTT and HTTP. MQTT is created for IoT but HTTP is very common in web-based systems and it is already used in other projects, so HTTP is selected for later use.

Chart 4. Potential technologies for the AIROS logger. The colors are indicating need of the technology. Green: needed, Yellow: possible, Red: not needed.



Name	Type	Between	Used	Must properties	Raspberry Pi	Arduino
PROFINET	protocol	🟩 gateway - robot	robot		yes	shield
Modbus	protocol	🟩 AIROS - gateway	AIROS	yes	yes	yes
Ethernet	technology	🟨 protoype - internet	no		yes	shield
RS-485	technology	🟩 AIROS - prototype/gateway	AIROS	yes	adapter	shield
Bluetooth	technology	🟥 AIROS - prototype	AIROS		yes	shield
WLAN	technology	🟨 protoype - internet	no		yes	shield
Cellular service	technology	🟩 prototype - internet	no	yes	adapter	shield
MQTT	protocol	🟨 protoype - server	no		yes	yes
HTTP	protocol	🟨 protoype - server	no		yes	yes

The AIROS logger can be connected to Internet with using either Ethernet, WLAN or cellular service which is the main connection way. After the communication to the Internet is established AIROS logger needs to send data to the server. The server has to be configured to work with a specific protocol. The AIROS logger system uses HTTP requests to handle data. HTTP request is used because of its simplicity and IT crew had experience of it. HTTP request are good and enough in small scale projects but more sophisticated methods are required if data packages are lost because of bad communication. Figure 12 shows communication links between devices.

The communication with AIROS can be done as Figure 13 illustrates. The figure shows that AIROS mainly communicates with Modbus, but it is also possible to control it with PROFINET (PROcess FIeld NET) which is a common protocol in industrial use. AIROS has also integrated Bluetooth but it isn‘t utilized in the AIROS logger system because there can be distractions between AIROS and the logger.

There are possibility to use Wireless personal area networks like Zigbee and LPWAN. They are great technologies for larger scale and connecting to near located devices. The use-case for using these kinds of technologies would be the following: AIROS loggers could be linked together, and it only needs one gateway to send data to server instead of every AIROS loggers have own cellular service. Our focus is to create working simple IoT devices that can work on its own so these technologies should be kept in mind. Implementing this kind of network would need tens of AIROS sanders in one location.

4 PROTOTYPE OF AIROS LOGGER

Today many industries rely on vertically specified machine-to-machine (M2M) solutions that involve extensive use of specifically customized hardware and software, which are typically designed only for that industry's service. Design, deployment, and maintenance of such proprietary solutions can cause high capital and operational expenditures. (Swetina i.e. 2014)

Building a complete IoT system is not always easy and it is important to choose the perfect hardware and software platform that is suitable for its purpose. There are factors, described below, that monitor our choice for the optimal platform for IoT system. The hardware platform has two computational concerns when it handles massive data flows, either decision making controller or data processing. An operating system is also a way to use the hardware platform and it helps developing an application with excellent performance. (El-Shweky i.e. 2018: 626)

The factors to help defining right hardware platform are:

1) Reduction of employed transistors, reflecting to die size, packing and unit cost.
2) Time to market – the markets require a generic solution to apply its demands.
3) Nonrecurring engineering cost – cost of the development process for software or hardware.

(El-Shweky i.e. 2018: 626)

Platform choice is based on the previous factor and there are two different types of platforms that could be chosen for prototyping.

1) Microprocessors

Microprocessors are used as platform for developing IoT devices and there are chips available which have a microprocessor with other blocks. The whole system is built on the chip with all its peripherals. The system acts as a gateway for the local devices on the

Internet and it is required to support several protocols for communication between devices, sensors and the Internet. There are many systems that are used commercially for this purpose. Arduino and Raspberry Pi are the most common. They are generic and could be used for many purposes. Hardware components are installed, which increases the power consumption, so it needs optimization for saving batteries. (El-Shweky i.e. 2018: 626)

2) Field programmable gate array, FPGA

It gives a less generic solution and provides a less time to market and nonrecurring engineering cost than application-specific integrated circuit (ASIC). FPGAs are expensive and their energy consumption is much higher than ASIC chips', which could be an issue for an IoT device. Combining FPGA and ASIC chips together gives advantages of both and FPGA's products are usually used for prototyping and testing the ASIC design that can be later mass product. (El-Shweky i.e. 2018: 626)

4.1 Raspberry Pi 3

Raspberry Pi was originally designed to schools for easy entrance to coding and developing. The first Raspberry Pi was released by Raspberry Pi Foundation in 2012 and it gained instant popularity because of low prize and overall its computer with all necessary components integrated to the board. There have been multiple models and now the Raspberry Pi is at version 3 which brought integrated wireless LAN and Bluetooth to the board. Version 3 also gained more power with quad core processor and it has multiple different inputs and outputs such as GPIO-pins (General-Purpose Input/Output), USB (Universal Serial Bus), HDMI (High-Definition Multimedia Interface) and Ethernet.

The Raspberry Pi is no longer a unique chip computer and there are lot of alternatives to it. Raspberry Pi has also a smaller and cheaper model, Raspberry Pi Zero with reduced features for example only two USB-ports of which the other is for power. Because we needed to work with port communication we didn't succeed to separate communication between the converter and USB-dongle so we used easier solution and chose Raspberry Pi 3 with all possible features.

4.2 Arduino

Arduino is an open-source electronics platform that consist of hardware and software. There are many different boards that are suitable and have different kinds of properties. Boards have potential to extend with shields giving them extra features.

All properties that are listed in Chart 4. are possible to achieve with a shield and Arduino needs at least two shields, cellular communication and RS-485. Arduino also has many ready-made libraries for example for Modbus communication.

4.3 Selected hardware

All the components are mainly designed for home use which means all the components aren’t best suitable for industrial use, but the main purpose is to show that you don‘t need expensive components to this kind of operations. Project did not have any precise budget but focus was to get affordable parts with good value and that all the parts could be changed with other similar parts.

Chart 4.**Error! Reference source not found.** has also small technology comprehensions between Raspberry Pi and Arduino which shows that products have differences in integrated communication technologies such as WLAN or Ethernet. Both products needs external accessories to use cellular or RS-485 communication but Raspberry pi can use plug-in USB accessories but Arduino needs specific designed shields.

The Chart 5. shows more details about the differences between prototype platforms. The selected prototype needs to perform actions fast, almost real-time, to capture communication between AIROS and robot. Arduino is fast to read sensors and other similar devices which are used directly through with digital or analogy ports. It is possible to capture messages through serial ports, but our focus is to rely on accessories that converts them readable form. Raspberry Pi is many times faster than Arduino and has more calculations speed to perform value converts, comparisons and database handling.

Chart 5. Differences between Arduino mega 256 and Raspberry pi 3.

	Arduino	Raspberry Pi
Core	8 bit	64 bit
RAM	8 KB	1 GB
Flash memory	256 KB	-
Clock speed	16 MHz	1.2 GHz
GPU	-	400 MHz
Programming	Arduino	python, C, C++, ruby etc.
multitasking	No	Yes
Network	-	Ethernet/WLAN/Bluetooth
USB interface	1 / connection to computer	4
Input voltage	7-12V	5V
Faulty shutdown	No effect	Possibility of files corruption
power consumption	~50 mA	2.5 A

The prototype should detect request and following response which is needed to remember so prototype needs some kind of database to remember read values. Both prototypes can handle this but it have to remember that Arduino has limited size of flash memory but Raspberry Pi uses external SD-memory card, which is much bigger.

Arduino has integrated flash memory that and it is designed for work when power is plugged-in and work until its plugged-out. Arduino doesn‘t have start up sequence and only the programmed code will be run and Raspberry Pi is small size computer that has start up sequence and there are lots of programs running all the time. There is possibility that if the Raspberry Pi isn‘t shutdown correctly it will corrupt files because it uses external SD-memory card.

Our choice of hardware platform, heart of the IoT-system, is a Raspberry Pi, which is very flexible for IoT and DIY-projects. The operating system Raspbian is based on Debian, so many Linux packages are compatible with it and without forgetting the huge number of active users. Selecting the Raspberry Pi to use instead of Arduino is based on that platform is enough powerful and it have built-in backup connection if cellular communication fails.

We used a USB to RS485 converter because they are so simple to use and they are cheap but randomly bought models may have some cons like erroneous communication. At the beginning of the project a couple of different USB to RS485 adapters were tested and they all seemed to work fine. Converters were bought from eBay and same models had different branding. We tested a PL2303HX chip-based model and another one was a FT232RL module and they both have automatic detection for data rate of the serial signal and they also support Linux, Windows and Mac computers. When the project was in testing we noticed that FT232RL model with AIROS logger program wasn‘t able to detect all inputs so we chose to use PL2303HX based converter. The problem was detection of the message and the converter was terminating signal detection, serial port program is able to identify message periods. So, for some reason converter couldn‘t detect correctly stop bits. Stop bits are used to detect the end of the Modbus message and they depend on what protocol and parity is used, for Modbus RTU (Remote Terminal Unit), which is used in this project, it is always 1 bit.

The choice of the cellular module i.e. dongle for outside communication was the next step. The module had to be able to function internationally, so it had to support as many frequencies as possible. Our 3G dongle had to support Linux and we looked at ones which incorporate their own network functionality. We chose to use a module by Huawei because it had a good reputation to work well with Raspberry Pi. Our device is Raspberry Pi model 3.

With these components it is possible to create an IoT device that collects the data from the AIROS. This device is illustrated in Figure 11.

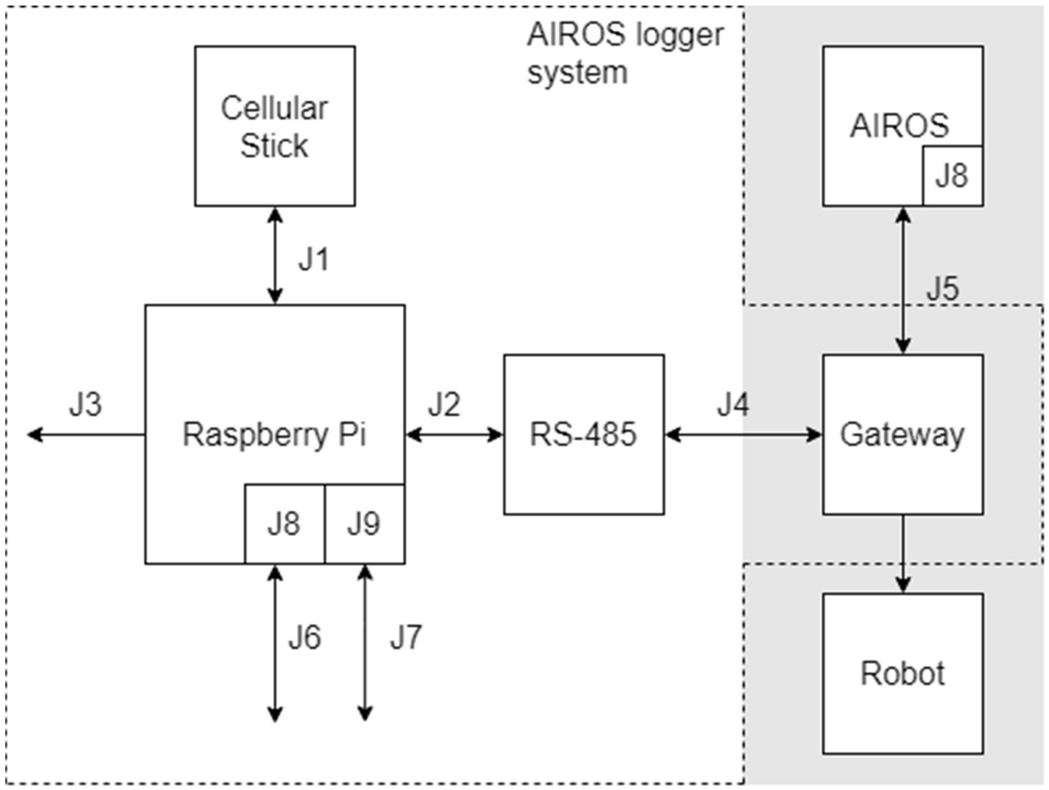


Figure 11. Installation diagram for the AIROS logger.

The Figure 11. illustrates the diagram of the AIROS logger that also contains wiring and available inputs and outputs for the logger. The figure has two systems, AIROS logger and the original AIROS sanding system that contains AIROS, gateway and robot. The AIROS logger system will be inserted to existing gateway is connected to AIROS with Modbus RTU RS-485 (J5). The Raspberry Pi is connected to cellular stick and RS-485 with USB-port (J1 and J2) and the devices uses 5 VDC power input (J3). Raspberry has able to communicate with Ethernet (J6), GPIO pins (J7), Bluetooth (J8) and WLAN (J9). The connections to the gateway is handled with Modbus RTU RS-485 (J4), there are three different pins A, B and ground. It is possible to that there is no gateway between AIROS and robot, then it is possible to use cable divider and connect directly to RS-485 cables to link the AIROS logger.

4.3.1 4G connection

Raspberry Pi has built-in Ethernet and WLAN, but AIROS logger has to be able to connect to the Internet by itself. Relying on Ethernet or WLAN could sometimes be easier or might be the only option if cellular coverage area is too bad to use. Our purpose of using a dongle is to build a so effortless solution that other companies give permission to use it and WLAN and LAN are usually highly forbidden to use excluding guest networks.

We need to send data from the AIROS logger to the remote server where our database is located. We also need to connect to the Raspberry Pi from the outside to update the device or just check something manually.

There are differences between 4G sticks that are in the market, but AIROS logger will use a Huawei E3372h-153. Using this 4G stick is easy because it only requires being inserted to Raspberry Pi and the installation of a modem software

4.3.2 Other used devices

Because the AIROS logger needs some data to monitor, it has to be attached to some place it can monitor the data exchange between two devices. In this case between the AIROS and robot which is moving AIROS. A good location for monitoring is the gateway which converts messages of different protocols to a suitable format for devices and all communication between them goes through it. The AIROS logger is connected to this gateway by an RS-485 adapter. It is possible to monitor the communication without a gateway and connect directly to AIROS by using cable divider like the Figure 11 shows.

4.3.3 Hardware flow

Hardware of our project consist of the previously mentioned devices: Raspberry Pi with RS-485-USB-adpter and 4G module accessories. The existing sanding entity includes AIROS, robot and computer which controls devices. The IoT-system is illustrated in Figure 12 which shows where connection of devices is shown. The AIROS logger will be inserted between AIROS and robot and then it will be able to detect communication between them. AIROS logger program will be running inside Raspberry pi which is connected to gateway with RS-485 converter. Raspberry Pi will be use 4G dongle to interact with the database and the server which enables remote access. Described hardware connection map can be seen in Figure 12. This figure only describes communication regarding the proto system and there might be some connections that are irrelevant. Inside the box all the data is transmitted in wires and outside connections go through the 4G dongle, which transfers data to wireless communication. Figure 12 illustrates connections when AIROS logger is not using VPN communication because VPN creates a tunnel to the server and all communication goes through that server. Now database dependency should go through the server and nothing else is changed.

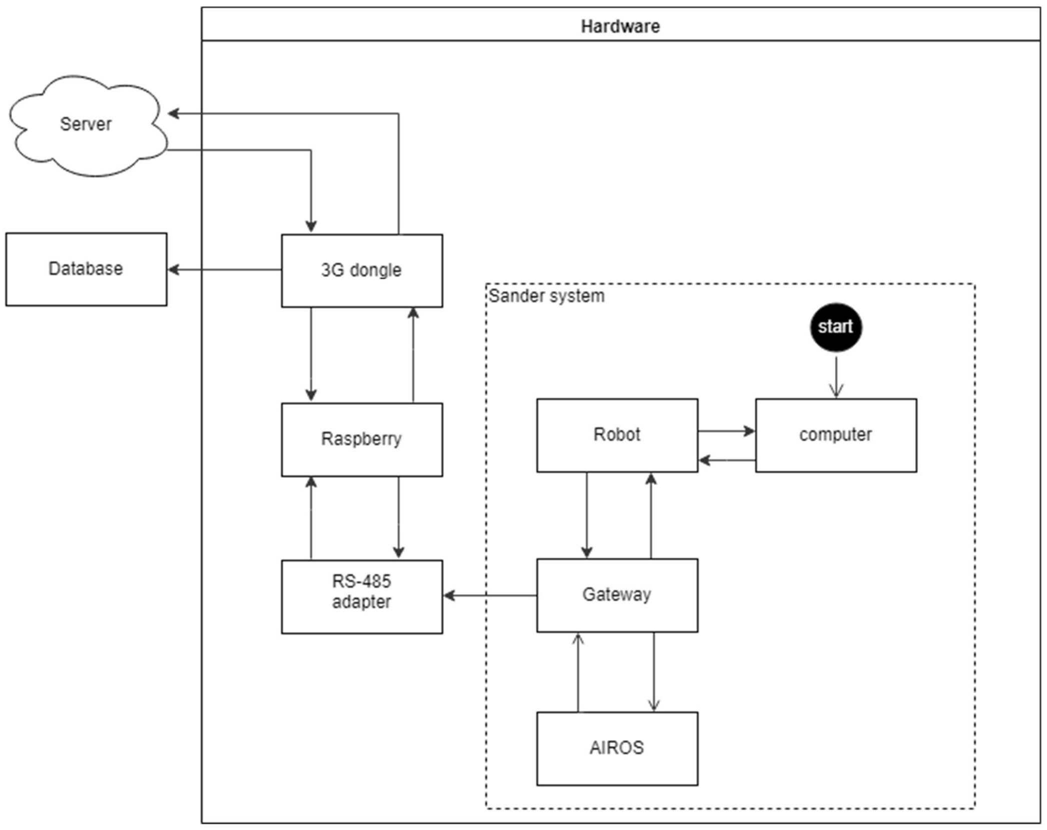


Figure 12. The hardware and communications of the AIROS logger system. This show how the hardware is bind to existing AIROS sander system.

The AIROS manual have described possible interface that user can use to communicate with the sander. The AIROS has three different interfaces: Modbus RTU, Profinet and Digital control. The main way to interact is with the Modbus RTU but it is possible to interact using optional Profinet gateway which converts Profinet to Modbus. Connections and interfaces to the AIROS is illustrated in the Figure 13. The figure is from the AIROS manual with more details on the connections and for example the following: AIROS motor drive uses 48 VDC power supply and Profinet can be used through the Modbus RTU gateway. (Mirka 2018b: 2-3)

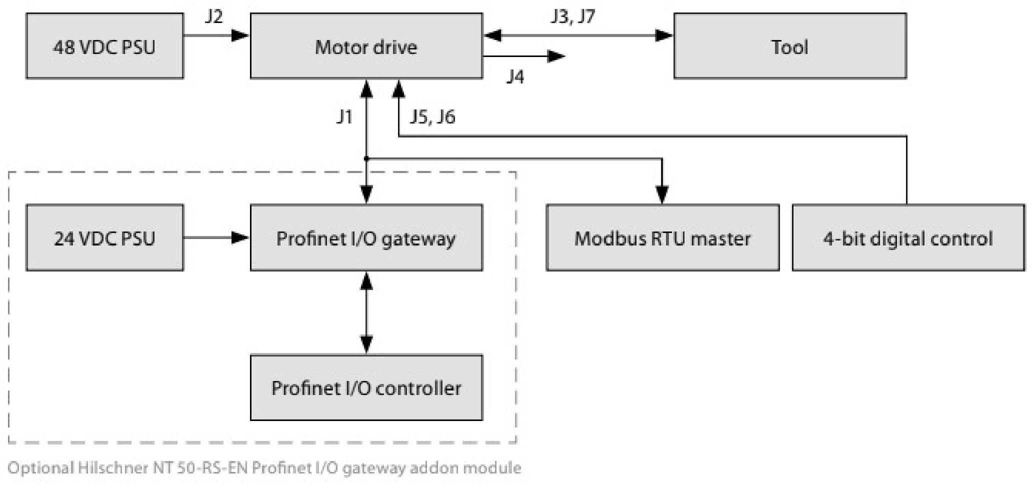


Figure 13. Installation of the AIROS sander. Markings are descripted in the below (Mirka 2018b: 2).

The connector that are illustrated in Figure 13. are following: J1 is Modbus connection; J2 is 48 VDC input; J3 and J7 are NTC (Negative temperature coefficient) thermistor connections; J4 is relay connections; J5 is for 4-bit digital speed control; J6 is for 15 VDC output.

4.3.4 Remote access

Raspbian OS has preinstalled remote access feature but it has to enable from the setting of the operating system. The remote access that Raspbian OS uses is SSH (Secure SHell) which creates secured connection between devices. Regular use of the SSH is connect device that has static IP address or local IP in local area network. Usually device is Connecting to remote device that is behind the NAT (Network Address Translation) which is technique to reduce IP addresses and all devices behind the NAT uses the same IP which gateway has. When the NAT is between devices it isn‘t possible to access directly to target device because its IP address. A workaround is to use server between them which has IP accessible IP address. Method is called reverse SSH which is connection that target device first creates a connection to the server and after it, it is possible to connect back to the target device using the same connection that already have created. This reverse SSH is illustrated in the Figure 14 where is steps 1-3. The first step is create connection between target device and server, green line. Second step is creating connection to the server from the source machine, blue line, and the last step is connect to the target machine using the same connection that is made in the first step, green line back to target machine. Meaning of the different lines in the figure are next: the green line is the connection that target machine is created and the blue line is the connection that user creates and the purple line in the server is mechanism that enables communication back to target machine.

The AIROS logger is using SSH and reverse SSH to create connection to the Raspberry Pi. Reverse SSH is used because 4G dongle uses NAT that blocks directly incoming traffic and device don‘t have static IP address what is needed for direct SSH connection. Figure 14 illustrates reverse SSH connection that is described before.

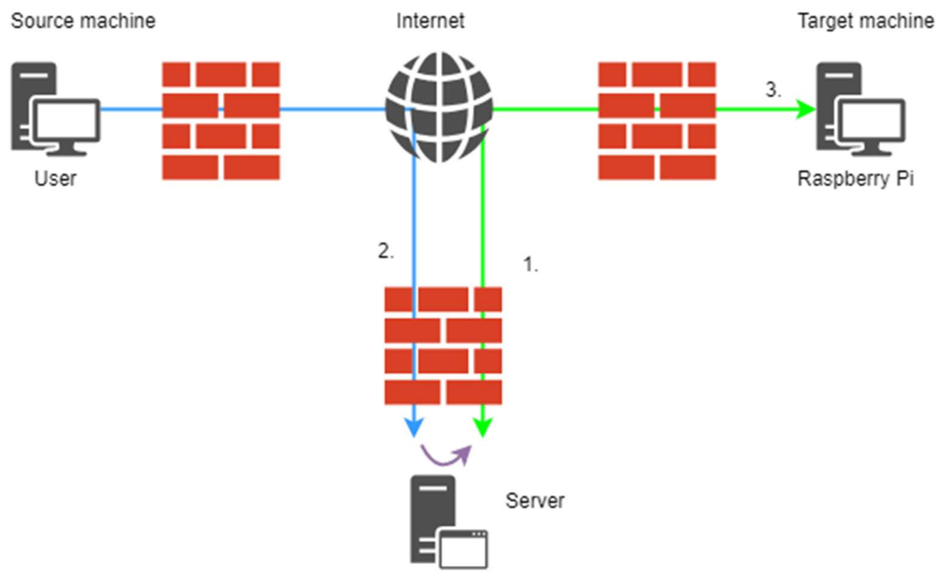


Figure 14. Reverse SSH connection.

The Internet connection of the AIROS logger is established with 4G dongle which means that outgoing traffic is working like it should, but incoming traffic back to the AIROS logger was problematic with regular cellular subscriber. New phone subscriptions are now using new IPv6 which makes it a little bit harder to create (stable) communication to the server. Subscribers NAT has only view incoming ports open and every subscriber has different ports open. For that reason the protype AIROS logger is using WLAN which gives reliable connection to the Internet and it is also possible to use reverse SSH and the AIROS logger doesn't have to care about blocked ports.

The alternative for the reverse SSH connection is VPN that removes the remote access problem when using 4G communication. Device network traffic goes through VPN tunnel and traffic goes out of the VPN server. VPN also creates a local network that can be used for SSH connection.

4.4 Software

The Raspberry Pi 3 is as any other platform that can use various operating systems (OS). The most popular choice and default OS for the Raspberry Pi is Rasbian which is based on Debian distribution which is Linux based. There are many alternatives for the OS, such as Ubuntu mate, Ubuntu core, Arch Linux ARM and Pidora. Ubuntu mate is based on regular Ubuntu, but Ubuntu core is designed for IoT and embedded devices and cloud computing. Chart 6. shows a couple of OS that are designed for Raspberry Pi and how they differ form each other. Most of the OS's are based on Linux but the Windows 10 IoT core is based on Windows and RICS of their own kind.

Chart 6. Operating systems for Raspberry Pi.

Name	Light	IoT	Maintenance	Linux	Comments
Rasbian	Yes	Yes	Official	Yes	Based on the Depian distribution, also headless.
Ubuntu MATE	No	No	Community	Yes	Full Linux with only MATE desktop
Snappy Ubuntu Core	Yes	Yes	Company	Yes	Almost same as Rasbian OS.
Windows 10 IoT Core	Yes	Yes	Company	No	Needs Windows 10 and Visual Studio
RISC OS	Yes	No	Community	No	Not Linux or Windows
Pidora	Yes	No	Community	Yes	Based on the Fedora distribution, also headless but not so active.

There are multiple Linux based operating systems that are quite similar to each other and have only small differences. Some Linux OS's can be used fully headless, without graphical user interface, and most of the using is through remote control which uses only a command line. The AIROS logger can be operated through the command line and the device needs less power and have better performance if we use headless OS.

Because of the Linux OS's are quite similar but the OS needs to have good long-term, device and program support and active community. We choose to use Raspbian in this project because it had all the criteria and it is the official operating system for Raspbian Pi.

The choice of the programming language was easy because Raspberry Pi supports Python well and it has many different libraries that will support AIROS logger functions and accessories and controlling Raspberry Pi GPIO pins are easy. Other preinstalled coding languages in the Raspbian are C, C++, Java, Ruby and Scratch.

The project software uses the object-oriented Python programming language. The project runs version 2 of Python, because some useful libraries aren‘t compiled to the newest version 3. The advantage of Python is its power and clear syntax. (Python 2019a)

The protype program takes advantage of Python's wide package library and uses them for many purposes, but interesting packages are PyModbus, pySerial and SQLite. PyModbus is a Modbus protocol implementation and it is easy to get basic AIROS information. PySerial encapsulates the access for serial port which enables monitoring the communication (Liechti 2019). SQLite is a C-language library that implements a SQL (Structured Query Language) database engine and it is small, fast and reliable and it is also the most used database engine in the world (SQLite 2019).

4.4.1 Software requirements

Mirka didn‘t give any special requirements for the AIROS logger software but a target was given: it needs to monitor Modbus communication and send changed values to an external server. The planning of the software started from writing down some requirements that AIROS logger software needs to fulfil:

1. Send Modbus message to AIROS to get device information.
2. Detect Modbus messages, request first then the response of the device.
3. Convert Modbus response data to a readable form.
4. Check if response is changed and hold the value.
5. Send changed value to external database.

After the listed requirements, simplified process structure is illustrated in Figure 15.

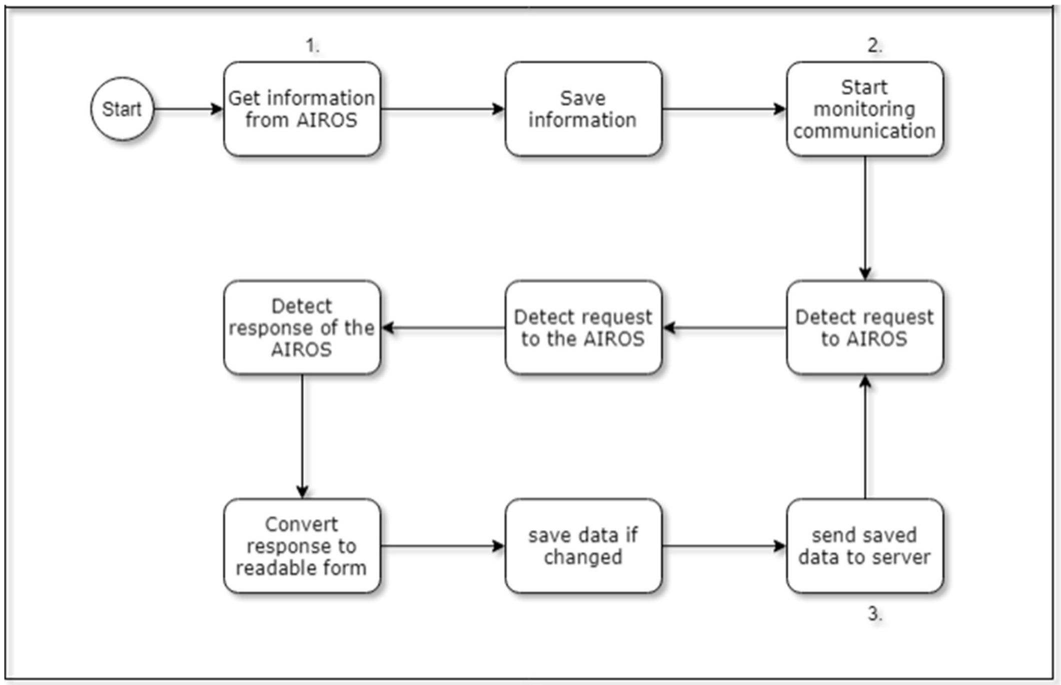

1.
2.
Start
Get information from AIROS
Save information
Start monitoring communication
Detect response of the AIROS
Detect request to the AIROS
Detect request to AIROS
Convert response to readable form
save data if changed
send saved data to server
3.

Figure 15. Simplified process structure of AIROS logger program.

4.4.2 Software flows

The AIROS logger program can be divided into a couple of sections as presented in Figure 15. First it sends Modbus request to the sander which will give basic information about the AIROS device. It will give data that isn‘t so useful for controlling the device, such as the device name and firmware version. Secondly it monitors the serial communication and saves the monitored data to the local database. Thirdly, it retrieves data from the database and sends it to the external server.

The software architecture wasn‘t directly planned but system was divided by functionalities.

1. Modbus communication
2. Serial communication
3. Response detection
4. Database service
5. Http service
6. Calculations / Modbus data checking

The Modbus communication takes care of the Modbus commands to and from AIROS. Serial communication handles the monitoring and gives Modbus a message that is even a request or response. The Response detection will detect if the Modbus message is a request or a response and after that it gives the data for further handing. The calculation / Modbus data checking will handle Modbus data and sends it to the Database service. Database service is able to build a database where the given data will be stored. The service is also responsible for retrieving and deleting the data from the database. The retrieved data is used in HTTP service which will send the data to an external service.

Figure 15 shows the first part of the software, requesting information. This part is Modbus request-response operation and saving the retrieved data in to the database. The high-level presentation of the fully functional code is illustrated in Figure 16. The meaning of the code is next: Application starts by going to Modbus service, which handles all the Modbus related functions. The service starts by making connection to the Modbus serial adapter and trying establish the connection to the AIROS sander. Without success, process will end. After successful connection, the program will create model that holds response values from the AIROS. Then data requests are send and responses are handled. The AIROS responses data is needed to convert to readable form and after that the data will be stored to object. It will be returned to main program after all responses are handled. The main program will store the values to the database.

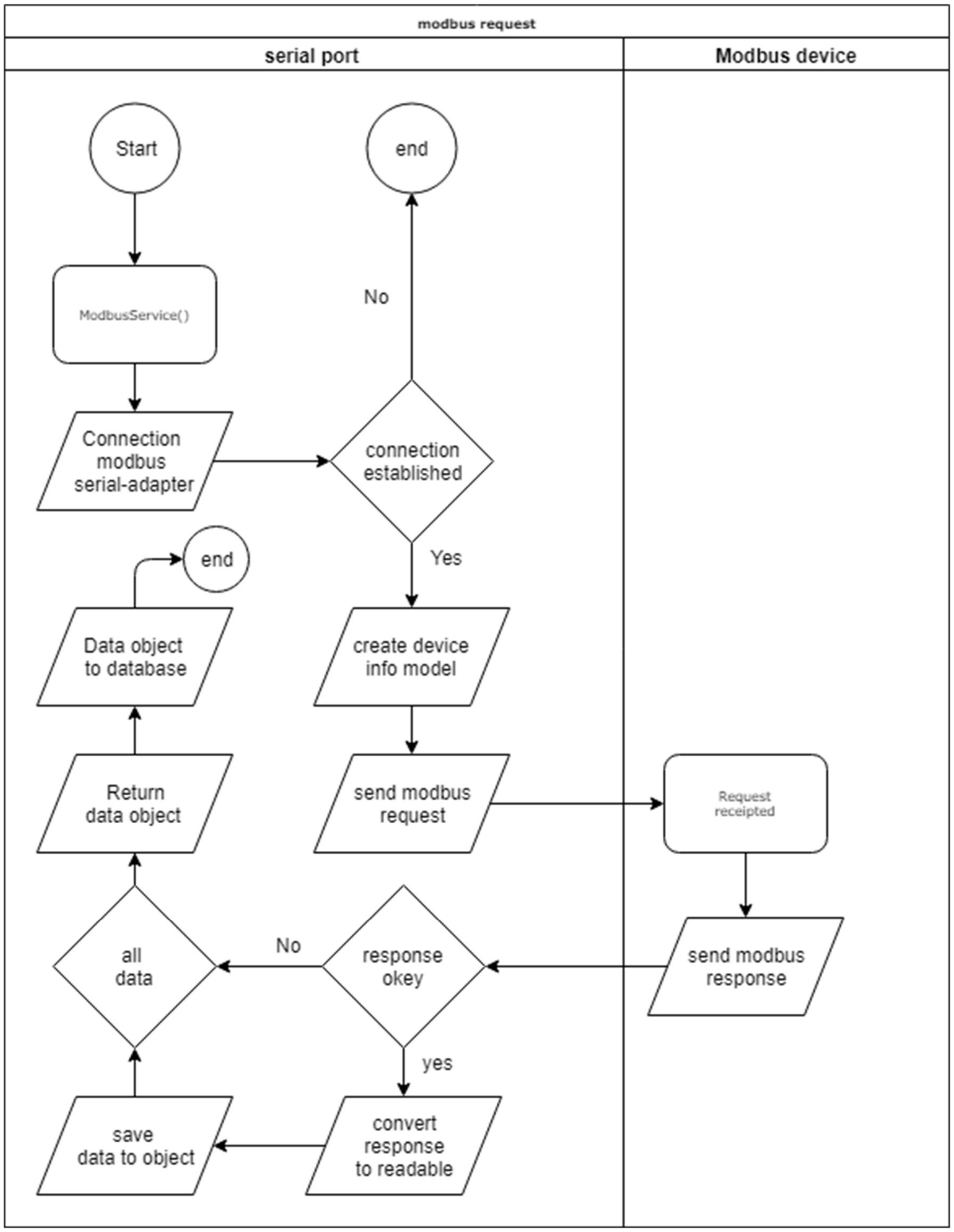


Figure 16. Modbus request and response function to AIROS sander.

Second part of the code that can be seen in Figure 15, monitoring the serial port communication, is part of the eternal loop. The loop will keep monitoring forever and this functionality is described in Figure 17, which illustrates data flow of the monitoring part and how the data is stored to the database.

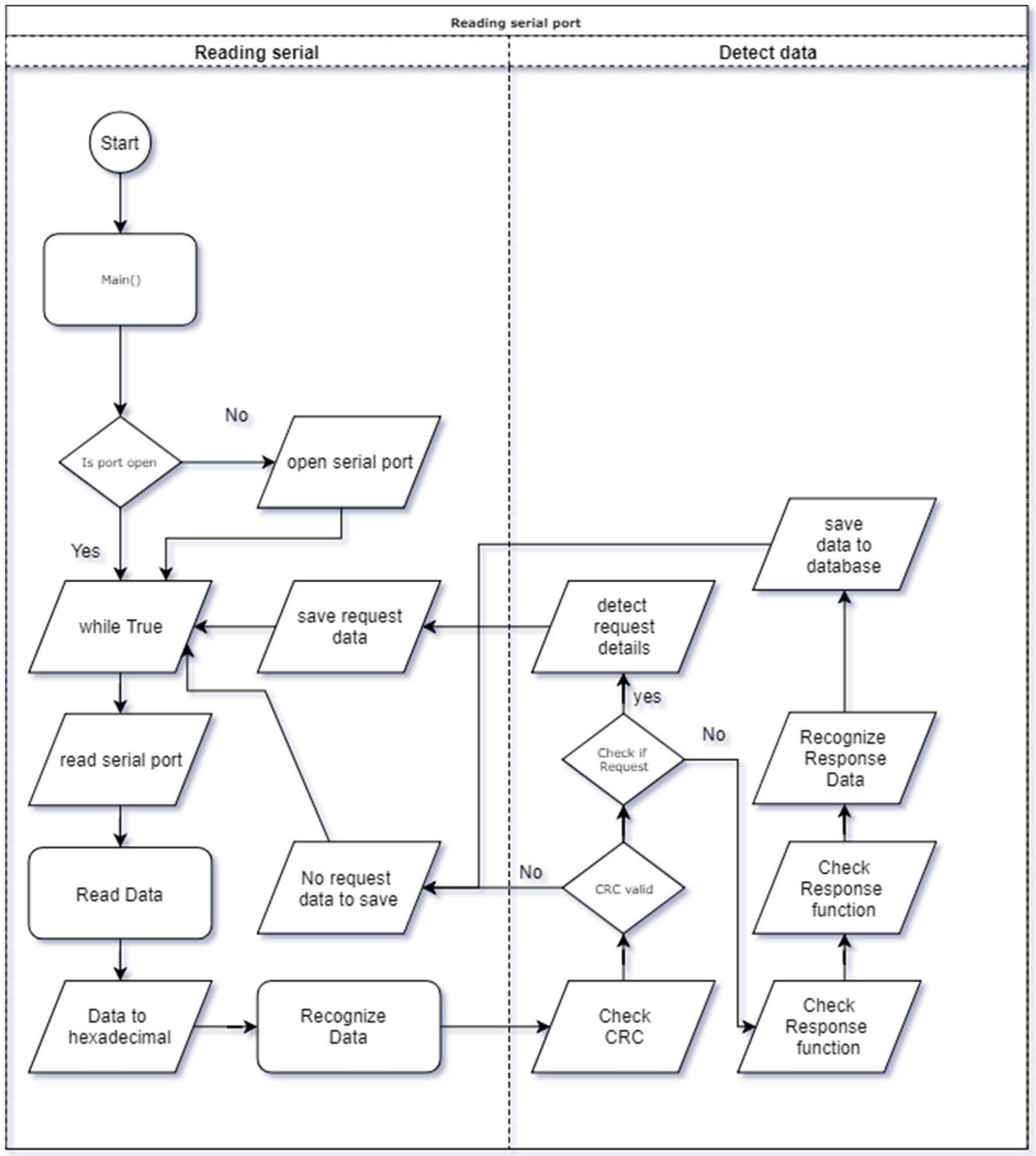


Figure 17. Process flow of AIROS logger and saving.

Part three of the code (Figure 15) describes a retrieving database data to data array which is transformed to multiple http-post requests. The request will send logged data to Mirka's server where it will be better protected and safe. The figure also shows that if the response from the server is successful, only that data will be deleted. Otherwise it will stay in the database. Figure 18 illustrates the dataflow of the third part of the program.

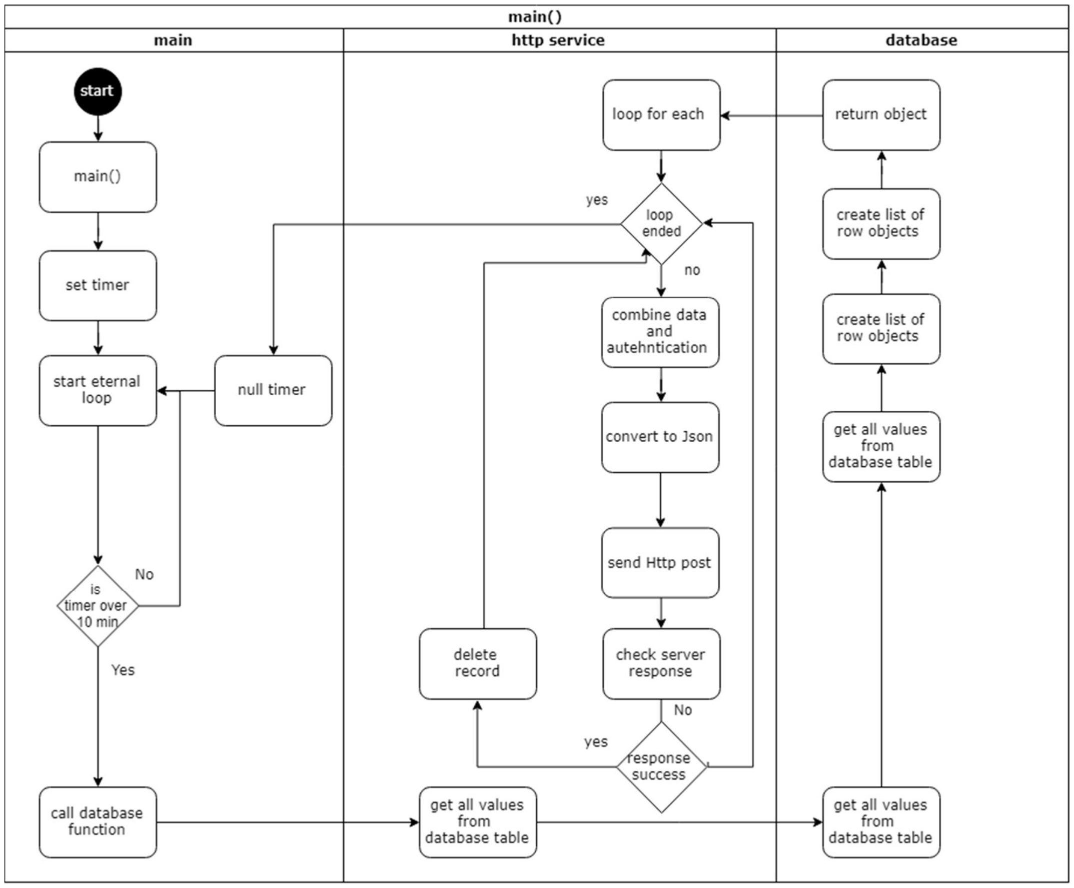


Figure 18. Dataflow of sending data to the external server.

4.5 Database implementation

Raspberry Pi has a local database and it is done by SQLite which is a very popular and light SQL database. It can have one database which has multiple different tables with different structures.

The structure of the local database isn‘t straight given but server-side database has almost the same values as AIROS and all the server-side values were typed to the string. Chart 7. illustrates a part of the server-side database structure.

Chart. 7. This is database structure is used in the Mirka server.

index	logger id	date	time	serial number	device name	part version	firmware version	usage time hours	usage time min.
1203	12034	18.2. 2019	18:51: 27	131411	AIROS 8095933	AI1.0 12345	1.21 Jan 3 08:58	804	54

In the beginning Raspberry Pi database was designed to be simple and the structure would look like Modbus structure with addresses and a value for them. This kind of structure gives an easy access to evaluate what has changed. **Error! Reference source not found.** illustrates the structure of the table and it includes only a couple of different columns: date, register number i.e. address and value. The register number is the Modbus register or coil number which depends on the Modbus’ function from 1 to 49999. Some interesting values are combined from different register numbers such as device name so Chart 8. styled Modbus like database structure was too much work for later use.

Chart 8. First Database version of the AIROS logger, where the database structure is modelled to look like Modbus message. Single address is single value.

index	Value	Timestamp	Date	Time	Address	Logger id
1	A	1555494151	04/17/2019	9:42:00	40001	132456
2	I	1555494379	04/17/2019	9:46:00	40002	132456
3	R	1555494379	04/17/2019	9:46:00	40003	132456
4	O	1555494379	04/17/2019	9:46:00	40004	132456

Previously database was modified to look like a database where are mixed variables and addresses. Now variables were used for values that needed combination. For example the device name was stored to one variable instead of 10 different addresses. This database was working well enough but the naming policy needed some clarity. The addresses were difficult to read as they were and needed renaming after for example the name of the device. This database structure is illustrated in **Error! Reference source not found.** 9.

Chart 9. Second database version of the AIROS logger, where database structure is also modelled to look like Modbus, but it can store multiple values in single address. Address is like variable which stores multiple single values.

index	value	timestamp	date	time	address	logger id
11	25	1555494151	04/17/2019	9:42:00	pcb temperature	1322456
12	AIROS 80959339	1555494379	04/17/2019	9:46:00	Firmware version	1322456

The server of the Mirka company can take only one row of parameters at time. The row can contain one or multiple values at the same time but if the AIROS logger sends only one parameter, it will change other row values to NULL. This makes it harder to use data if only a part of the data is shown and other data is missing. It would be simpler to have all the possible data even if they have not changed.

The current database which have value per row will need investigate unchanged values to fill caps and combine them in the end, when the data is sent to the server.

The logging program was doing unnecessary combining it was easy to see where optimization is needed. The Database was modified to keep all current values in one row and now whole row is ready to send to the server. Now the database structure is the same as the server-side database which can be seen in Chart 7. This design is needed for reducing the time and work for sending data to server and it will simplify the code structure.

At the beginning of the logging we will save MAC address to the database which is also for identification at the Mirka server. With this identification we can update the data without getting multiple rows.

It is rational to keep all current values in stash and also keep log what have changed and when. So, the database is made of two different tables one for current values and one for logged values. When the current value is changed it will automatically be added to the logged table. This copying is done with SQLite triggers, which are handy to automate simple tasks. This logging table can be erased when needed so only content of this table is sent to external Mirka server and if sending of the values successful they will be deleted from the data-table.

5 SOFTWARE DEVELOPMENT

The software is developed in Python language, so it can be compiled and run in different OS and when a source code is running for the first time the source code is compiled to byte code which is a faster way to run than a source code is. AIROS loggers software is coded with using Microsoft Windows OS and Visual Studio Code - code editor which also Microsoft product. Code editor has many useful extensions that make coding easier. In the source code developing progress Lint is used for finding syntax and style problems. There are some differences between how different OSs show attached USB-devices but overall it is easy to write code in one and use it in another one.

There is no perfect code because there are always something to improve but it is important to aim for a better code. One way to improve the code is using Lint and using testing frameworks for testing the written code.

5.1 Unit testing

In the AIROS logger software development process, any other software testing extensions or any behavior-driven develop framework like Jasmine weren’t used. Jasmine is commonly used testing frameworks for JavaScript language and it has also been developed for Python based web projects (Pivotal Software 2019). Python has its own unit testing framework, which is inspired by Junit, testing framework for Java (Python 2019b). Because unit tests catches mistakes and exceptions it is important use them.

The AIROS logger software uses Python’s own test framework to test the whole programs code. There are multiple files that had to be tested and most of the functions are tested in some level. Python’s own test framework doesn‘t have code coverage but approximately 70% of the functions have been tested to exclude third party code. After starting to use unit tests it found some bugs that were hard to find and that were not considered.

5.2 Developing the software

After using the perfectly functioning dummy messages it was time to run the code with some real data. The easiest way to get real data is to use a Modbus request to the AIROS board. The board has pre-set information such as serial number and device name. It also has an integrated temperature sensor that gives real data. The Modbus request is used for developing functions that converts and checks Modbus messages.

The monitor functions cannot be developed correctly if the data isn‘t coming from the serial port. Using AIROS with sampler, which continuously requests different values, creates opportunity to capture real Modbus communication. The captured data from the communication has to be checked and modified to readable format.

One of the first problems encountered during the serial port communication was the detection of the Modbus messages. Using randomly selected Modbus adapter caused problems with detection of the correct Modbus messages. This problem appeared first randomly when assembling testing/developing board. Messages were assembled wrong almost every time which made the messages look as if they were cut and joined randomly. After changing the serial adapter to another corrected the messages. So even the bad adapter worked fine to show the messages. It didn‘t notice when the message ended and that for it wasn‘t suitable for monitoring a task.

The Modbus is a request-response type of communication, protype software needs to know the previously detected Modbus message. Software has to know what the robot is requesting so it can read response messages data and store it correctly.

The form of the database is important and if the database is formed rationally it is easy to maintain. Also saving and cueing the data is fast to execute. The structure of the database was changed after the first program generation. At first the data was saved by using the Modbus address and one row of data consisted of logged data, logging date, Modbus address code and unique identification for AIROS logger. This kind of structure is good for inserting values because it is simple to add or update only one value. When all data is scattered it needs to combine or fill the missing values before sending them. These functions make it more complex and it needs to work harder to create JSON-message that is needed for exporting data to Mirka’s server.

The database structure was modified to keep all the possible data within the database row, so JSON-message is now simple to create without extra work. This structure also keeps all current values, so it simplifies the data preparation at data analysis stage which means there are less data gaps to fill.

One of the concerns was saving the data to the database so that it would keep all the desired data and clear it after the data is processed. The first program version consisted of two separate threads for data monitoring and logging values and for sending logged values to the server and saving the data again if sending was faulty. The idea for this threading was to separate the fast and slow functions so that monitoring would continue logging and other thread handles process of sending the data which might cause latency. Threading is causing some database problems because only one database connection can be written to the database at same time, but it is possible to read in multiple threads or programs. This problem was bypassed with creation of another database.

In the next program version, threading wasn‘t used anymore. This was done because using one database is much simpler and clearer because also the entire code had permission to edit the database if needed. Overall the new program version had no significant delay in data sending process, so it could be used. SQLite has triggers which is used when program is inserting or updating values to the main database date, which keeps record of the current values. Triggers are programmed to automatically “copy-paste” changed row to the other table that contains all the logged rows. This another table is cleared when sent to Mirka’s external server.

Modbus communication has always one master and one or multiple slaves that give information if requested. In the system, AIROS is acting as Modbus slave and the robot as master and when AIROS logger is installed, it also acts as Modbus master. This means there will be two master devices in the same system. Usually Modbus RTU system has only one master, but it is possible to add several master devices for example with gateway. AIROS logger isn‘t much communicating with AIROS, but it is possible that both robot and logger communicate at the same time. Logger is communication only when the device, Raspberry Pi, is started. The device is only requesting some unusual parameters that might not been added to be asked such as device serial number, name or firmware version. This overlapping request may cause problems but probably they don‘t do any harm to the system or the connected devices. The request that AIROS logger sends is just requesting of the parameters, so in the worst-case scenario a wrong device is responding.

5.3 Protype testing

When the software was coded it was simultaneously tested in small scale, but proper tested wasn‘t run. Testing the whole program would require some real and changeable data to investigate how the system behaves.

5.3.1 Target

The purpose of the test is to investigate how the software will react to real data that might be corrupted and if the software is powerful enough to capture all the messages. Also how the collector device will react to other devices with different cases.

5.3.2 Setup

The testing system will be Raspberry Pi version 3 with Modbus converter and AIROS circuit board with gateway sampler. This sampler will request continuously and systematically all AIROS values. In the later AIROS board and sampler is replaced with real AIROS and robot which won‘t differ from this case because AIROS board has integrated a temperature sensor, which gives changeable data to log. The software is modified to send logged data every 10 minutes to the server and delete that data from the local database.

5.3.3 Cases

Testing plan 1 is next: Raspberry Pi is connected to the gateway between AIROS and sampler which is already requesting information. Raspberry Pi is turned on and it should request some information from the AIROS and then it should start monitoring the communication between the AIROS and the sampler. This testing is planned to take approximately 20 minutes and it should send data at least two times to the server.

Testing plan 2 is next: All the devices are turned off and the Raspberry Pi is first turned on and then other devices are turned on. Raspberry Pi shouldn‘t have been able to get any Modbus response from AIROS so how monitoring and sending logged data now works. This testing is planned to take approximately 15 minutes and it should send data at least one time to the server.

Test plan 3 has almost the same specs than test plan 1 has but the environment of the AIROS is cooling down after the start. Cooling is done by inserting ice under the AIROS, so the temperature sensor should react to the temperature drop.

5.3.4 Results

At first checking the logs of the software showed that some messages aren‘t detected correctly even though the bad RS-485 converter wasn‘t used. Serial port was detecting messages automatically, but it won’t identify messages 100% correctly but overall it detects most of them. The messages are randomly broken but sometimes they appear a lot but then there can be a long time between two broken messages. This kind of distortion can be caused by a bad connection or wires. This kinds of distractions were detected in every test case.

The second test case caused major errors for the program and the program stopped working after it tried to send a request to the AIROS and the response never returned. The software got an undefined variable which crashed the program. The software was fixed and tested again without any errors.

After the third test, the system was detecting changes and values were logged to the local database and sent to Mirka’s server as it should. The test was successful.

Overall, tests showed that AIROS logger software is working fine with the Raspberry Pi and RS-485 accessory. The AIROS logger detects almost all serial port communication and it tries to detect every message, even if they aren’t Modbus messages or just shattered. AIROS logger software is focusing on Modbus protocol so it needs to detect a request message before a response. So if there is for example some noise between request and response or the response is shattered, it won’t detect the response at all even if the messages are fine.

5.4 Data analysis

After the testing cases, the IoT-system was attached to the working AIROS sander and robot. The data logging was performed in the laboratory of Mirka in Jeppo. The IoT-system was left to gather the logging data from AIROS and this data got approximately one week of irregular usage and most of the data it got in a couple of different sections. The data was stored in approximately 3700 data points, and this gave a small hint on how the sander is being used. The data analysis chapter is describing what kind of data is stored and how to read the gathered data. For reminding, the purpose of the AIROS logger is to collect usage data that can be used for improving the AIROS and detecting need of service.

AIROS or AIROS logger program won‘t detect any information on the sanding material or sanding paper, it gives only usage statistic and sensor data. AIROS has two different temperature sensors, one in the motor and another in the PCB (Printed Circuit Board).

Figure 19 shows the sensor data from the motor and PCB and how much the device is heating up when the load changes.

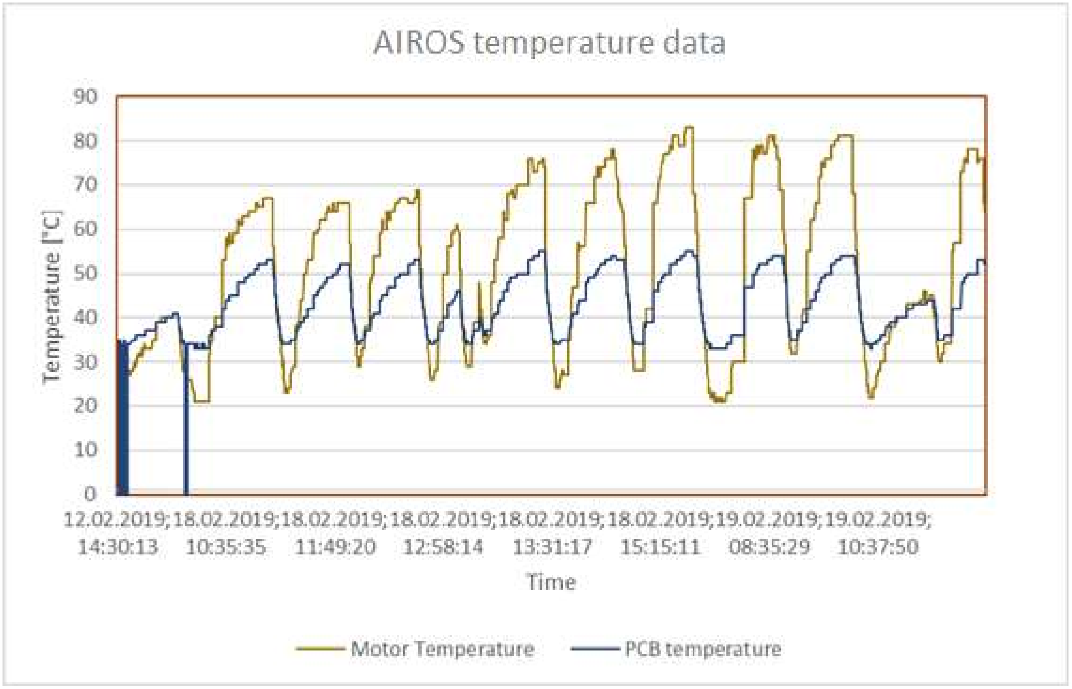


Figure 19. AIROS temperature data from inbuild motor and PCB temperature sensors. Data illustrates the heat that sanding progress produces.

Sensors are in different spots and the motor temperature increases faster because the motor is generating heat more and faster and the PCB is located further from the heat source. When machines are gathered enough data, different patterns can be detected that indicates different needs or things for example when motor temperature is always too high which can indicate need for motor service.

In Figure 20 an average current and speed changing during the same period is shown. The target of the speed is usually 10000 RPM (Revolutions Per Minute) which a sander reaches fast. The speed seems often almost static but current level is changing which is caused by the load of the sander. The needed amount of power for keeping the same speed depends on the force, sanding material and sandpaper and for example polishing glass needs less power than sanding wood.

There can be found some patterns when examining the Figure 20. The first and second last bar can mean lot of things. For example, AIROS is sanding corner of the target item or that sander is partly off from the target item.

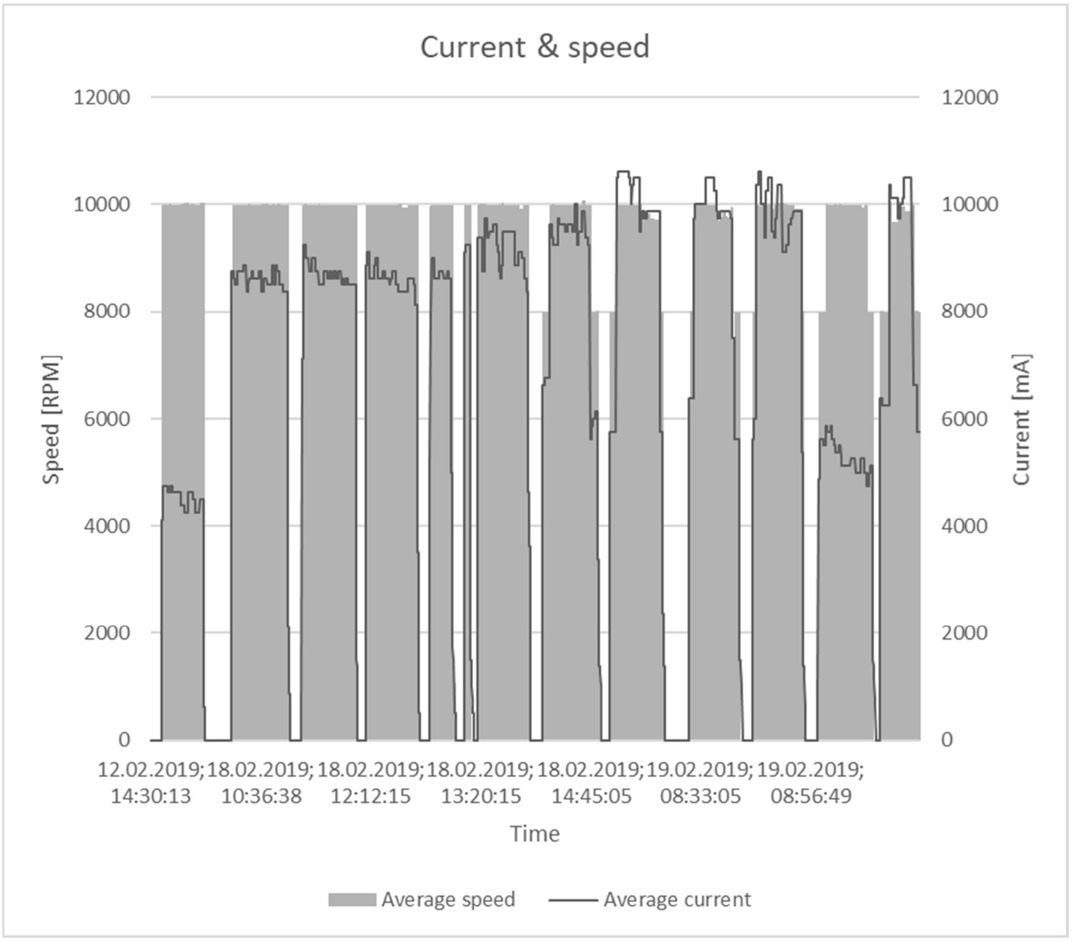


Figure 20. AIROS average speed and current, where can be detected pattern when it approaches and recedes from the sanding piece.

Previously shown average speed (Figure 20.) and temperature data (Figure 19) is combined in Figure 21 which shows how temperatures are connected to the current. Using AIROS with specific load will heat up especially the motor and graphs are surprisingly same with a small gap. This similarity is a result of the load which heats up the motor because it needs to use more power. Meaning of the power is the rate of doing work which means either physically something is moving or generating heat. In this small sample the produced heat can be said to be directly proportional to the current of device.

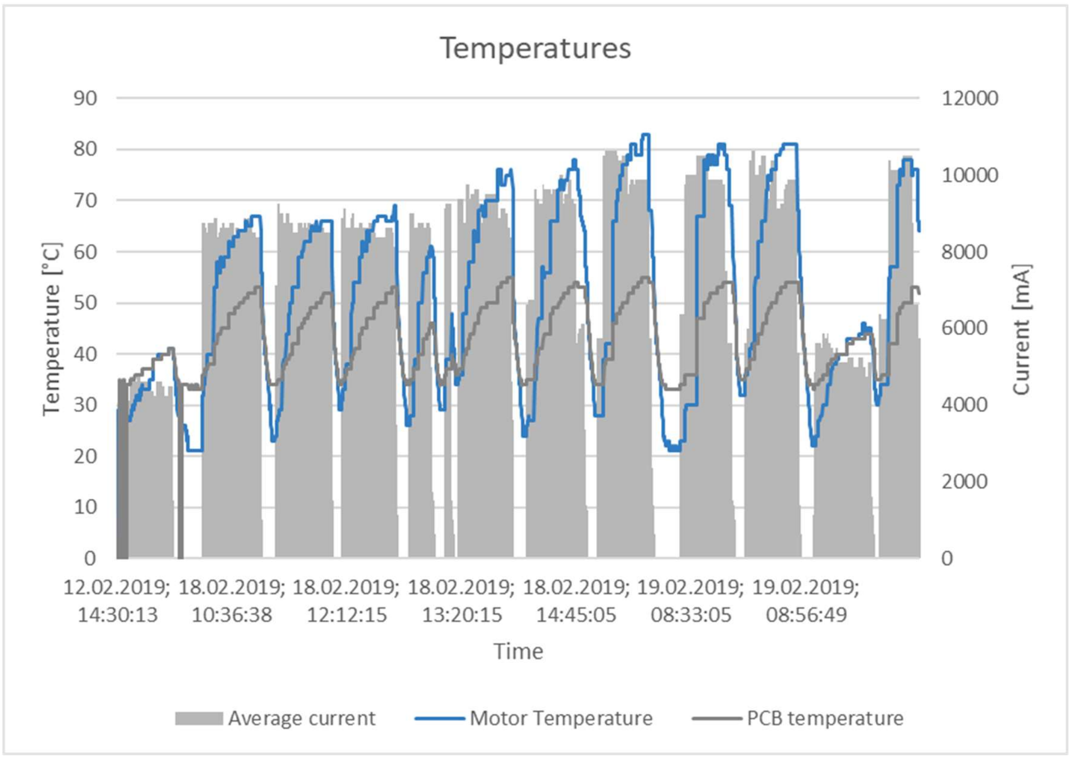


Figure 21. AIROS mixed temperature and current data that illustrates that produced heat is directly proportional to the current of device.

6 CONCLUSION AND FUTURE WORK

The goal of this thesis was to create an IoT device i.e. AIROS logger that can autonomously monitor communication between AIROS and the robot. AIROS is acting as Modbus slave and the robot as master which means there will be two master devices in the same system, because AIROS logger is also acting as master. Usually a Modbus RTU system has only one master but it is possible to add several master device which is not very common. AIROS logger isn‘t communicating with Modbus almost at all and only once when AIROS logger is starting. The device is only requesting some rare parameters that aren‘t so important for a sanding process such as device serial number, name or firmware version. This may cause problems but probably it will not do any harm to the system or connected devices and in the worst case some other connected device is responding.

The AIROS logger is detecting all the communication that comes from the serial adapter and the AIROS logger program is made for detecting requests which follow responses. A program is made for detecting only communication between specific devices, AIROS and the robot. If the communication has some irrelevant communication such as requests/responses from other devices or anything else a serial port registers. This irrelevant communication is probably disturbing the AIROS loggers logging because a Modbus message has the address of the target device whose detection isn‘t included in the software. This program related issue should be concerned to fix in the next stage.

The AIROS loggers Raspberry Pi is fine most of the time, but it can be unused for days. Raspberry Pi is a card computer and like computer as any other, sometimes they need rebooting when getting stuck or too slow. One of the purposes was to create communication between the AIROS logger and service which is done by using HTTP-post method. This HTTP-get is very old and generates huge amount of waste traffic which is expensive if data usage is limited or priced by usage. Then it could be reasonable to use MQTT or similar communication to get better quality, performance and smaller data post by devices. This change could be done in the future.

The AIROS logger was supposed to work smoothly with 4G dongle but communication back to the AIROS logger encountered some issues that were subscription related. Subscriber is blocking Internet communication with NAT and some ports are closed. The cellular subscription that is tested with don‘t have static IP address so AIROS logger needs reverse SSH that’s ables connection back to the device. The AIROS logger solutions are made for specific cellular subscribers so if subscription provider is changed it need some reconfiguration. Instead of using the reverse SSH connection it is recommended to use VPN to pass subscription related issues.

The AIROS logger with the monitoring program isn‘t perfect and overall it needs more long-term testing because it is supposed to work with any human interaction and now it is generating randomly small errors over the time that are critical for operating the program.

Mirka has myMirka software that is a graphical interface for end-users that see the data from the Mirka power tool devices. The AIROS isn‘t available for all users yet because the data isn‘t sent to its database. So using this existing software database could add value for users and Mirka which for example doesn‘t need to maintain different databases. In the end this AIROS logger needs some small adjustments and will from the Mirka to get AIROS logger from prototype stage to mass-product.

REFERENCES

Advantech B+B SmartWorx, (2019). *SmartSwarm 351*. [online]. [7.3.2019]. Available from Internet: http://advantech-bb.com/product-technology/iot-and-network-edge-platforms/smartswarm-351/

Aldowah, H., Rehman, S., Umar, I., (2019). Security in Internet of Things: Issues, Challenges and Solutions, Advances in Intelligent Systems and Computing, Pages 396-405. Volume 843. ISBN: 978-331999006-4

Amodu, O. A., Othman, M. (2018), Machine-to-Machine Communication: An Overview of Opportunities. *Computer Networks*. 255-276. Volume 145. Elsevier.

Andrianto, V. C., lam, J., Widodo, R. N. S., Lee, S., Lee, H., Lim, H., (2018). Toward implementation of oneM2M based IoT platform. *Journal of Theoretical and Applied Information Technology*. Volume 96. 418-425. ISSN: 1992-8645

Anttalainen, Tarmo. (2003). *Introduction to Telecommunications Network Engineering*, Artech House, ProQuest Ebook Centra

Atop Technologies, (2019). *Modbus Gateways*. [online]. [7.3.2019]. Available from Internet: http://www.atop.com.tw/atop/product/product_detail/data/atop_iapl/en/modbus_gateways/mb5901_series/

Bahashwan, Abdullah & Manickam, Selvakumar. (2019). A Brief Review of Messaging Protocol Standards for Internet of Things (IoT). pp.1-14.

Bluetooth Special Interest Group (2016). Bluetooth core specification [online]. [1.6.2018]. Available from Internet: https://www.bluetooth.com/specifications/bluetooth-core-specification

Brooks, J. (2018). Latest oneM2M IoT standardisation progress to be showcased at MWC 2018. [online]. [12.1.2018]. Available from Internet: http://www.onem2m.org/news-events/news/174-latest-onem2m-iot-standardisation-progress-to-be-showcased-at-mwc-2018

Bziuk, W., Phung, C. V., Dizdarević, J., & Jukan, A. (2018). On HTTP performance in IoT applications: An analysis of latency and throughput, 2018 41st International Convention on Information and Communication Technology, Electronics and Microelectronics (MIPRO), Opatija, pp. 350-355.

Carey, T. (2017). oneM2M Work on IoT Semantic and Data Model Interoperability. [online]. [5.5.2018]. Available from Internet: http://www.onem2m.org/news-events/onem2m-in-the-news

Chris Liechti (2019). *Welcome to pySerial's documentation?* [online]. [12.1.2019]. Available from Internet: https://pyserial.readthedocs.io/en/latest/index.html

El-Shweky, B. E., El-Kholy, K., Abdelghany, M., Salah, M., Wael, M., Alsherbini, O., Ismail, Y., Salah, K., AbdelSalam, M. (2018). Internet of things: A comparative study. Conference in Las Vegas: 2018 IEEE 8th Annual Computing and Communication Workshop and Conference (CCWC). ISBN: 978-1-5386-4649-6

Geng, H. (2016). *Internet of Things and Data Analytics Handbook*, John Wiley & Sons, Incorporated. ISBN: 9781119173649

Haidine, A., El Hassani, S., Aqqal, A., El Hannani, A. (2016). The Role of Communication Technologies in Building Future Smart Cities, *Smart Cities Technologies.* IntechOpen. ISBN: 978-953-51-2808-3]

ionSign, (2019). *Gluon GMU491 Cloud Gateway*. [online]. [7.3.2019]. Available from Internet: https://ionsign.fi/en/iot-big-data/gmu491/

Jorge L. Olenewa, (2012). *Guide to Wireless Communications.* ISBN: 978-1111307318

Kayal, P., & Perros, H. (2017). A comparison of IoT application layer protocols through a smart parking implementation. *2017 20th Conference on Innovations in Clouds, Internet and Networks (ICIN)*, Paris. pp 331-336. ISSN: 2472-8144.

Kuzin, Mikhail., Y., Shmelev, V., Kuskov, (2018). [online]. [20.5.2019]. Available from Internet: https://securelist.com/new-trends-in-the-world-of-iot-threats/87991/

Larmo, A., Ratilainen, A., & Saarinen, J. (2018). Impact of CoAP and MQTT on NB-IoT System Performance. *Sensors (Basel, Switzerland)*, vol. 19

Mekki, K., Bajic, E., Chaxel, F., Meyer, F. (2018). A comparative study of LPWAN technologies for large-scale IoT deployment. *ICT Express*. Elsevier BV. ISSN: 2405-9595

Minoli, D. (2013) Building the Internet of Things with IPv6 and MIPv6 : The Evolving World of M2M Communications, Edition 1, Publisher: Wiley. ISBN: 9781118473474

Mirka. (2018a*). Mirka home-page.* [online]. [16.9.2018]. Available from Internet: https://www.mirka.com/fi/fi/-Top-Menu-/Mirkasta/

Mirka. (2018b*). Mirka AIROS, Electrical manual.*

Oliveira, G. M. B., Costa, D. C. M., Cavalcanti, R. J. B. V. M., Oliveira, J. P. P., Silva, D. R.C., Nogueira M. B., & Rodrigues M.C. (2018). Comparison Between MQTT and WebSocket Protocols for IoT Applications Using ESP8266, *2018 Workshop on Metrology for Industry 4.0 and IoT,* Brescia, pp. 236-241. ISBN: 978-1-5386-2497-5

Oliveira, Luiz, Rodrigues, J., Kozlov, S., Rabêlo, R., Albuquerque, V. (2019). MAC Layer Protocols for Internet of Things: A Survey. *Future Internet.* ISSN 1999-5903

OMA specWorks (2019). *OMA SpecWorks FAQ* [online]. [7.2.2019]. Available from Internet: https://www.omaspecworks.org/about/oma-specworks-faq/

oneM2M (2018). *Functional Architecture.* TS-0001. p. 535

Patzke Robert (1998). Fieldbus basics. *Computer Standards & Interfaces.* Volume 19, Issues 5–6, p. 275-293. Elsevier. ISSN: 0920-5489

Pivotal Software, (2019). *Using Jasmine with Python* [online]. [13.2.2019]. Available from Internet: https://jasmine.github.io/setup/python.html

Ptiček, M., Čačković, V., Pavelić, M., Kušek, M., Ježić, G., (2015). Architecture and functionality in M2M standards. Conference in Opatija, Croatia: 2015 38th International Convention on Information and Communication Technology, Electronics and Microelectronics (MIPRO)

Putera, C. A. L., & Lin, F. J., (2015). Incorporating OMA Lightweight M2M protocol in IoT/M2M standard architecture, *2015 IEEE 2nd World Forum on Internet of Things (WF-IoT)*, Milan, pp. 559-564. ISBN: 978-1-5090-0365-5

Python Software Foundation (2019a). *General Information* [online]. [12.1.2019]. Available from Internet: https://docs.python.org/3/faq/general.html#have-any-significant-projects-been-done-in-python

Python Software Foundation. (2019b). *Unit testing framework* [online]. [13.1.2019]. Available from Internet: https://docs.python.org/3/faq/general.html#have-any-significant-projects-been-done-in-python

Quinnell Rich (2013) *Vertical vs. horizontal: Which IoT model will thrive?* [online]. [17.11.2018]. Available from Internet: https://www.embedded.com/electronics-blogs/other/4422131/Vertical-vs--horizontal--Which-IoT-model-will-thrive-

Rawat, P., Singh, K. D., Bonnin, J. M. (2016). Cognitive radio for M2M and Internet of Things: A survey, *Computer Communications*. 1-29. Volume 94. Elsevier. ISSN: 0140-3664

Reynders, D., Mackay S., Wright E. (2004). *Practical Industrial Data Communications: Best Practice Techniques*. Elsevier Science & Technology. ISBN: 9780750663953

Sohraby, K., Minoli, D., Occhiogrosso, B., Wang, W. (2017). A Review of Wireless and Satellite-Based M2M/IoT Services in Support of Smart Grids. *Mobile Networks and Applications.* Volume 23, Issue 4, pp 881–895. Springer US. ISSN: 1572-8153

SQLite (2019). *What Is SQLite?* [online]. [12.1.2019]. Available from Internet: https://www.sqlite.org/index.html

Swetina, J., Lu G., Jacobs, P., Ennesser, F., Song, J. (2014). Toward a standardized common M2M service layer platform: Introduction to oneM2M. *IEEE Wireless Communications*. Volume 21. Issue 3. 20-26. ISSN:1536-1284.

Tsiatsis V., Fikouras I., Avesand S., Karnouskos S., Mulligan C., Holler J., Boyle D. (2014). *From Machine-to-Machine to the Internet of Things: Introduction to a New Age of Intelligence.* Elsevier Science & Technology. ISBN: 9780124076846

Upadhyay, Y., Borole, A., Dileepan, D. (2016). MQTT based secured home automation system. *Symposium on Colossal Data Analysis and Networking (CDAN)*. Conference in Indore, India. ISBN: 978-1-5090-0669-4

Waher, Peter (2015). *Learning Internet of Things*, Packt Publishing Ltd. ISBN: 9781783553532 pp. 278