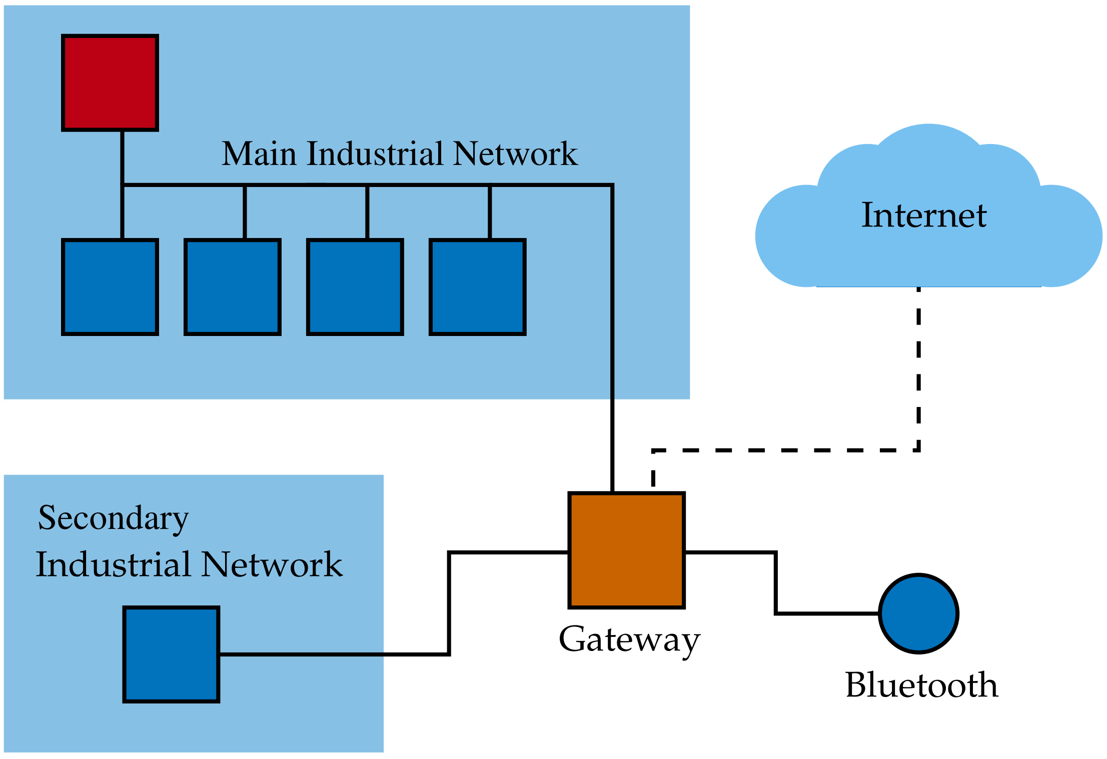
CHALMERS
UNIVERSITY OF TECHNOLOGY

UNIVERSITY OF GOTHENBURG



Profinet Industrial Internet of Things Gateway for the Smart Factory

Master's thesis in Embedded Electronic System Design

HENRIK JOHANSSON

Department of Computer Science and Engineering
CHALMERS UNIVERSITY OF TECHNOLOGY
UNIVERSITY OF GOTHENBURG
Gothenburg, Sweden 2017

Master's thesis 2017

Profinet Industrial Internet of Things Gateway for the Smart Factory

Henrik Johansson

Department of Computer Science and Engineering
Chalmers University of Technology
University of Gothenburg
Gothenburg, Sweden 2017

Supervisor:
Endian Technologies AB: Andreas Henriksson
Chalmers: Lena Peterson, Department of Computer Science and Engineering

Examiner:
Chalmers: Per Larsson-Edefors, Department of Computer Science and Engineering

Master's Thesis 2017
Department of Computer Science and Engineering
Chalmers University of Technology and University of Gothenburg
SE-412 96 Gothenburg
Telephone +46 31 772 1000

Typeset in LaTeX
Gothenburg, Sweden 2017

Profinet Internet of Things Gateway for the Smart Factory
HENRIK JOHANSSON
Department of Computer Science and Engineering
Chalmers University of Technology and University of Gothenburg

Abstract

The Internet of Things revolution in industrial production environments imposes new requirements on devices, hardware and software in the production systems. In previous systems, each vendor could have their own proprietary protocols and platforms but as more and more devices and sensors are connected together, increased collaboration and standardization is necessary.

This thesis aims to aid in building the bridge connecting an industrial system and the outside world, but also on solving an existing problem in industrial networks regarding the set up for new devices. The thesis also aims to give a compilation of protocols and platforms that can be used in industrial applications. A practical solution to bridging the networks and solving the problem is done by implementing an industrial network stack (Profinet) in a gateway device. The resulting prototype gateway can communicate with conventional network traffic as well as the industrial network protocol Profinet, with both master and slave functionality.

Keywords: Industrial Internet of Things, IIoT, Industrial IoT, Industrie 4.0, Industry 4.0, Profinet, Gateway

Acknowledgements

I would first like to thank my splendid supervisor Lena Peterson at Chalmers University of Technology for standing up with my mailbombs, broken sentences and misspellings.

I would also like to thank my supervisors Andreas Henriksson and Peter Malmberg at Endian Technologies AB for their ideas, technical support and guidance.

Finally I would like to thank all other colleagues at Endian for a warm welcome and friendly work environment.

Henrik Johansson, Gothenburg, June 2017

Contents

List of Figures

List of Acronyms

ASCII: American Standard Code for Information Interchange
AWS: Amazon Web Services
BLE: Bluetooth Low Energy
DDS: Data Distribution Service
FPGA: Field Programmable Gate Array
GNU: Recursive acronym for “GNUs not Unix”
GSD: General Station Description
HTTP: Hypertext Transfer Protocol
HTTPS: Hypertext Transfer Protocol Secure
I/O: Input/Output
IDA: Intelligent Distributed Automation
IIC: Industrial Internet Consortium
IIRA: Industrial Internet Reference Architechture
IIoT: Industrial Internet of Things
IP (Copyright): Intellectual Property
IP (Networking): Internet Protocol
IRT: Isochronous Real-time
IT: Informational Technology
IoT: Internet of Things
LAN: Local Area Network
MQTT: Message Queue Telemetry Transport
OBD: On-board diagnostics
OS: Operating system
OSI: Open Systems Interconnection Model
OT: Operational Technology
PAN: Personal Area Network
PLC: Programmable Logic Controller
PaaS: Platform as a Service
RT: Real-time
SSL: Secure Sockets Layer
SaaS: Software as a Service
TCP: Transmission Control Protocol
TLS: Transport Layer Security
UDP: User Datagram Protocol
VLAN: Virtual Local Area Network
WAN: Wide Area Network

1
Background

The fourth industrial revolution, so called Industrie 4.0, is ongoing [1]. Manufacturing companies are digitizing their entire value chain, factories and machines get connected, and data analysis is essential in what is called the Industrial Internet of Things (IIoT) [1, 2]. New technologies are emerging such as network-connected sensors, 3D printing and new tools for the operator such as augmented reality and collaborative robots [1]. An important business potential can be found in helping companies to introduce wireless technology in production systems, to ease the introduction of new sensors and to provide information to and from the factory.

1.1 Introduction

A major hurdle for implementation of the IIoT, is the industry's reluctance to embrace the new technology [3, 4]. The reluctance has been problematic because it is currently necessary to make changes in the already established system which is time consuming and expensive. In large industrial systems the time to implement new functionality is even longer, which leads to long expensive down-times. The down-times can in turn cause companies to update their systems even more infrequently, resulting in aging devices that are even more difficult to replace. How do we break this negative spiral?

One way to prevent the spiral is to make it easier to update the devices, such as PLCs and input/output (I/O) nodes. One such way is to decentralize the work with more dynamic modular nodes. In this thesis project a gateway will be created between the outside world and the industrial network (Profinet) [5, 6]. Profinet will be combined with the Intelligent Distributed Automation (IDA) interface, without modification to existing hardware or architectural layout, see Figure 1.1.

IDA is a system, created by the company SEVAB, that is designed to create a plug-and-play functionality making it possible to add, move or remove a Programmable Logic Controller (PLC) unit [7, 8] without any reprogramming [9]. However, there are still issues during the setup. When a new node or PLC I/O device is introduced to the system, the controller PLC will need to account for all possible hardware configurations that could be used by the new node.

To make the network more scalable, we could introduce a new level of abstraction by implementing the gateway between the new node and the controller PLC. As the

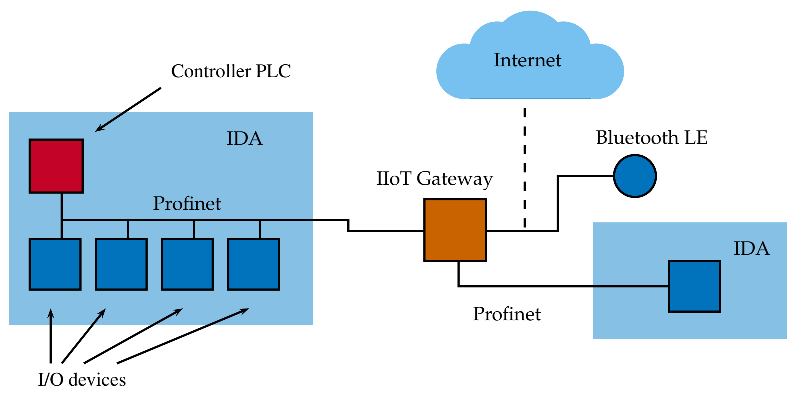


Figure 1.1: Overview of how the gateway can be connected to different systems, such as to another Profinet network, to a Bluetooth device or connected to a cloud service.

controller PLC only needs to see the gateway as a new PLC node with a fixed set of hardware, the tedious setup work concerning the controller PLC is reduced. See Figure 1.2 and 1.3 for illustration.

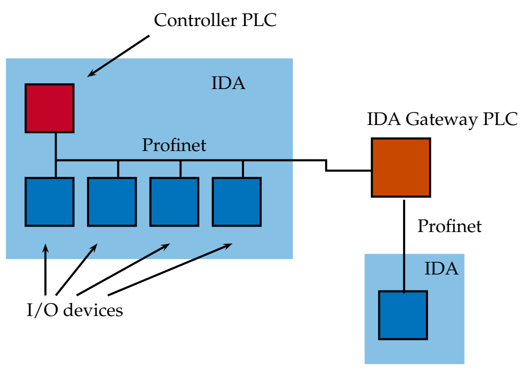


Figure 1.2: Network layout for a proxy service

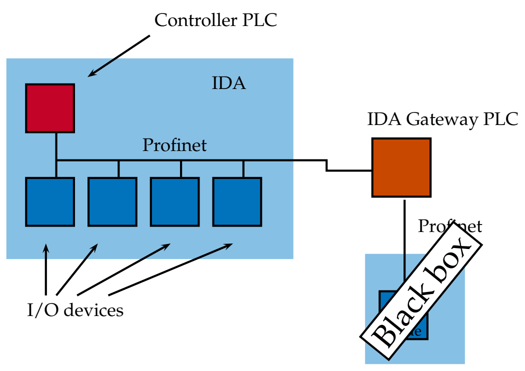


Figure 1.3: Network as seen from the controller

Another, easier, way to see the dynamic features this gives is to imagine that there are two blocks. The top one is always the same and only communicates with the Profinet IDA interface, and the bottom one is whatever function that is desired. The desired function can be the proxy service, a network connected IoT gateway or simply communication with a Bluetooth Low Energy (BLE) device [10], see Figure 1.4.

Additional feature, other than the proxy service, is the ability to connect to the outside of the industrial network, for example to a wireless BLE device. The gateway is connected to the Profinet network on one side and can have any number of technologies on the other side. By combining the gateway with the IDA system, it is possible to create a flexible production with reduced changeover times. At the same time, information about factory status will be available outside the industrial network, for example to a cloud service. There are already available cloud services

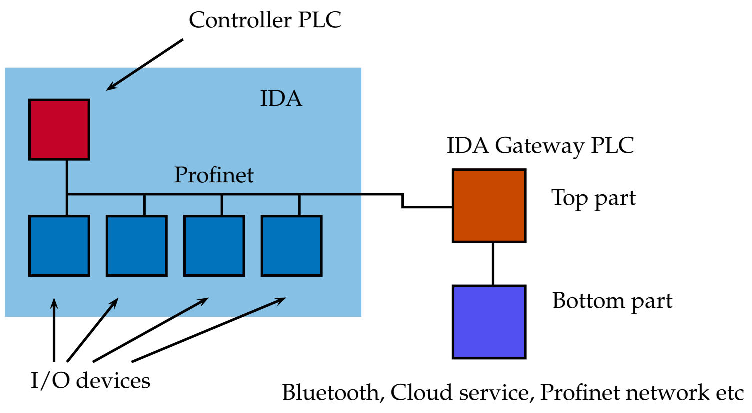


Figure 1.4: How the gateway combined with different technologies can be seen as two blocks

for Profinet, such as Proficloud [11], but none with the IDA protocol.

1.2 Goal and Purpose

The work is carried out at Endian Technologies AB, and their two main objectives are to:

- Evaluate what the Industrial Internet of Things (IIoT) is and any predictions on the direction it is headed in the future, both as an overall guideline but also focusing on where Profinet fits into the future.
- Develop a proof of concept gateway that connects to a Profinet network on one side and to an arbitrary service on the other. The key focus is to provide a dynamic gateway that can be used to connect Profinet devices to other, previously unavailable, services.

The overall goals of this thesis project are to:

- Evaluate what other industrial network protocols, other than Profinet, used in industry that could be used in the thesis project and what protocols to expect in the future.
- Identify the security and ethical aspects of having the gateway connected to a cloud-based service.
- Develop a demonstrator that is connected to the industrial network (Profinet) at one side, and to sensors or a cloud platform at the other. The connected sensor should use Bluetooth and serve as an wireless emergency stop that workers can carry with them.
- The gateway should also be able to implement the proxy service shown in Figure 1.2 and 1.3, by connecting the Profinet network and a PLC node to elevate the abstraction level for setup of new nodes.

1.3 Limitations

The project focus is to connect a gateway to the Profinet network, and although research will be done about other industrial networks that could be used, no testing or implementation of other networks than Profinet will be done. Only hardware that run a GNU/Linux system will be evaluated to use for the gateway, i.e. no Windows Embedded or other proprietary devices will be taken into consideration. Profinet will be implemented purely in software, no external chips or soft-cores in FPGAs will be used. The constructed gateway should only serve as a proof of concept and not a finished product to be used in industry.

1.4 Ethical Aspects

The biggest concerns found during this thesis project would be regarding the cloud service. If you yourself (the company) do not have control over the server that runs the service, there is no guarantee that the information only will be used in your best interests. If one company is responsible for all factories cloud service, they could either leak information to other competing businesses or worse, security details to other countries. But having all services connected to one cloud service also make it more vulnerable as the interest to crack the security might be higher. Another issue that might arise is if the cloud service has sensors connected, such as cameras or microphones, the workers could be under surveillance.

Another concern would be regarding implementing a commercial Profinet stack, as the software would not be “free”. Here meaning free as in “freedom not free beer” [12]. By implementing a commercial stack, there are some restrictions that one has to follow. The stack for the slave side is given in C code, which gives better transparency. The master side, however, is run as an application and can be considered as a black box, which makes it difficult to see what other instructions is executed other than that of the Profinet drivers.

In the project a prototype wireless emergency switch is implemented which has some ethical concerns regarding whoever is responsible if this does not work correctly. Additionally by implementing the proxy functionality in between two Profinet network, another concern can be raised if the gateway does not forward security and safety critical messages in time.

1.5 Thesis report outline

This paper starts with some background information on industrial environments, such as programmable logic controllers (PLC) and how to program them, some information about industrial security and industrial hardware.

Next, some information will be given about Internet of Things and the industrial variant, with some of the most used protocols and platforms and a short section about SSL/TLS encryption. Following that is a chapter about network models and how conventional traffic is different from the industrial variants.

In the Gateway design chapter, the demonstrator design decisions and implementation choices are explained and evaluated, and finally the results and conclusions are given.

2 Industrial environment background

This chapter covers information regarding industrial environments, such as hardware, software and programming but also security concerns.

2.1 Programmable Logic Controller (PLC)

A programmable logic controller [8], or PLC for short, is a computer designed for industry and is used in the automation of the processes in a manufacturing system. PLCs were designed to replace earlier technologies used, such as relays and timers. They are made to withstand harsh environment such as large temperature deviations, excessive vibrations, high humidity and dusty conditions. The logic controllers are used to control timing-critical operations such as milling, lathing, drilling and printing, they are hard real-time systems with deterministic timing [8].

2.1.1 PLC programming

There are a number of different ways to program a PLC, and five languages for programming languages and programmable control systems are defined in the standard IEC 61131-3 [13]. Two of the five languages are text-based, whereas the rest are graphical. The two text based ones are Instruction List (IL) and Structured Text (ST). The graphical ones are called “Sequential Function Chart” (SFC), “Function Block Diagram” (FBD) and “Ladder Diagram” (LD) [14], where the latter (LD) is the one used in the project, and is the only one that is being evaluated.

2.1.2 Ladder Diagram (LD)

Ladder Diagram originated as a way to represent the layout of relay logic [15]. It is a graphical way to program, which is visually similar to how an actual electric circuit looks like. A basic function can be seen in Figure 2.1 with its equal electrical schematic drawing in Figure 2.2.

In Figure 2.1 the input X, called “normally open input” in Ladder Diagram terms, is used to control the output Y, noted as a “coil”. The name “coil” can be somewhat confusing, and is a remnant from previous relay logic where you control a relay by activating a magnetic field in a coil to close the circuit. The output does not have to go to the coil part of a relay, but can be used to control anything. The result of the circuit is that with a high input, the output also goes high.

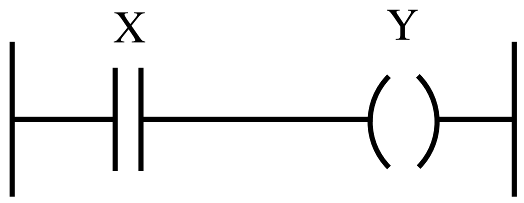


Figure 2.1: Ladder Logic for the simplest of schematic. A high input, X, gives a high output, Y.

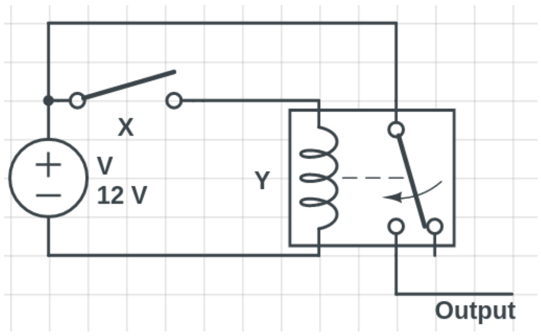


Figure 2.2: Electrical schematic for simple connection. A high input, X, gives a high output, Y.

The usefulness of having a graphical way to program the PLCs is mostly for engineers with an electrical background, as the layout is very similar to that of how actual hardware would be connected, but for software- or computer engineers, a text based variant is often more familiar. Another positive feature for simple designs, is the way to debug, as one can easily see what parts are high and low.

2.2 Industrial safety

To maintain a high level of safety in an industrial environment, there are a number of different systems and devices in place, such as emergency stops. Another safety approach is by equipping the machines and robots with sensors that activate when humans are nearby and either slow down the production or completely halt until the persons have left the danger zone [16].

Another view concerning personal safety is protective clothing, such as glasses, helmets, gloves and steel boots. A relatively new idea is concerning personal emergency stops where the workers themselves all have individual emergency stops that once pressed, stop the nearest machines [17]. This could be combined with some type of smart textile that senses if the fabric has been ripped or smashed, and the clothing would itself send the distress signal to halt nearby machines. Another approach where it might be easier to implement this type of emergency stop, is in industries with hardhat requirements. The hat might already have a battery to provide power for a headlamp, and the company would thus not be too foreign to having another battery and emergency stop installed. Having the emergency stop installed in the helmets also makes them distributed around the workplace and in the case that one stop does not work, another one can be located more quickly than if it would be a conventional wireless emergency stop. Another feature of the emergency stop would be to also alert if a worker has a dangerous health status such as a weak or non-existing pulse, and would then contact medics or other nearby workers, with the coordinates and the health status of the worker.

2.3 Industrial hardware

Other than the human safety part, there are other demands for industrial components such as the demands on the hardware. The main requirement of industrial grade hardware is to be durable. Once replacement is necessary, there should exist replacement parts even after the parts' end-of-life. The hardware should also be able to operate in harsh environments with large temperature fluctuations from -40 to +100 degrees centigrade [18], to have a low power consumption without active cooling such as fans, be able to sustain vibration and shocks [19], high humidity, corrosive liquids and electromagnetic interference [20].

3
Internet of Things background

This chapter covers basic background information regarding Internet of Things (IoT), both conventional IoT but also the Industrial IoT. Further some platforms and protocols are described with their different pros and cons, it should also be noted again that one of the main goals of the thesis is to evaluate and compare different platforms and protocols. Too understand the practical part of the thesis, meaning the gateway, it is only necessary to read the sections regarding Node-Red, OPC UA, MQTT and Redis. The other sections are meant for evaluation and comparison purposes.

3.1 Internet of Things

Internet of Things, or IoT, is the term for networking physical devices or "things" together. The IoT term is usually used for all types of "smart" or connected devices, however there are four main branches or areas concerning IoT. The branches are commercial, enterprise, consumer and industrial IoT [1]. Consumer is the one most people might think about when they hear IoT, which is devices used for personal applications, such as having the refrigerator notifying once the milk is soon to expire, or wireless speakers that sense in which room people are located. Commercial IoT is used by organizations to increase some business value such as marketing, Enterprise IoT is also used by organizations but the primary focus is not to make more money or increase profit, it is more about collecting information about process status and information gathering [21]. An interesting example of how collecting information and distributing it can be seen in smart visors designed to help firefighters. The helmets uses a "heads up display" (HUD) that shows the image from a thermal camera, the helmets will also be connected together so that both the firefighters and other personnel outside the buildings can get get an idea of where they are located relative to each other. Industrial IoT concerns connected devices in industrial and manufacturing environments [1, 2], more about this in the section 3.4.

3.2 Cloud Computing

Cloud computing is where resources, processes and data may be shared with remotely located servers, either in remote countries, cities or closer. A primary benefit of utilizing cloud computing is that the set-up cost is minimized as one does not have to purchase any physical hardware and servers, but instead rely on the supplier's hardware [22]. Another positive feature is that the computational power that a company or organization needs can be changed dynamically, certain months might have high demands whereas the next could be less and it is possible to compensate these fluctuations by changing how much hardware/computer power you use at the cloud servers. A negative side of using cloud computing is that the organization or company does not have total insight on what computations are run on the data, as they would if it was on one of their own servers. A similar type of network-based computing is fogging or fog computing [23]. Instead of having the servers located remotely, as in cloud computing, in fogging the servers are located at the edge of the organization network. Fogging does not help in the dynamical sense as with outsourcing cloud computing, however, it does make it more transparent to the organization.

Proficloud

Proficloud is a Profinet Cloud service created by Phoenix Contact [11]. The units, PCs, PLCs etc, are connected to the cloud service, Proficloud, and the traffic is encrypted using TLS [24, 25]. See section 3.10 for more information about SSL and TLS.

3.3 Industrie 4.0

Industrie 4.0 is a term that is often used in discussions about Industrial IoT and Industrial Internet. Industrie 4.0 is the name that the German government issued to their project of being “the leader of the industrial market” as both manufacturer and provider [26], hence the German word, industrie, for industry. The name is based around the three industrial revolutions that have occurred previously and we are now traversing into the fourth. The first revolution was in late 1700 to early 1800 when machines, such as the weaving loom and steam or petroleum powered devices, were introduced [27]. The second revolution was in late 1800 with production lines and mass production were initiated, and the third revolution was in the 80s with computer controlled automated robots. The fourth revolution is believed to happen now with smart devices such as actuators and sensors being connected together.

3.4 Industrial Internet of Things

Industrial Internet of Things, Industrial IoT, IIoT or Industrial Internet [3] are some of the names given to the technology of connecting industrial machines and devices together, henceforth called Industrial IoT throughout this report. Industrial IoT is geared towards connecting devices to add a higher level of abstraction that makes it possible to get an overview of everything in the organization. There already are methods, such as quarterly reports, used to give an indication of the current status of the organization. An improvement can be made by having the devices connected as the results would be more frequently updated and any abnormal data could be addressed right away. Another direction of Industrial IoT is to have sensors in concumable parts, such as bearings, brakes or lights. By having sensors in these parts and components, maintenance needs can more easily be predicted and avoidable down times or unnecessary replacements can be prevented. The sensors could also make it easier in remote locations, where it takes a long time for replacement parts to arrive, such as for wind turbines or boats far out at sea.

An extension to the idea of predicting how worn parts are and when to replace them is the concept of “virtual copies” or “digital twins” [28, 29], where actual physical hardware is given a virtual software based copy. The hardware sends data on how much it has been used and what parts have been changed, the digital version then updates its values and an overview of the hardware can be made without access to it.

Another term that comes up is Industry 5.0 [30], where some argue that Industrie 4.0 and Industrial Internet is about connecting the devices together, Industry 5.0 is more focused on the human-machine interaction and collaboration. Where humans can work alongside machines but also somewhat about augmented reality [31], where the operator gets graphical illustrations through a transparent screen which make it possible to depict images and objects as they would in reality. These types of features makes it easier for workers with assembly and repairs. Imagine that instead of having a somewhat confusing technical documentation of how to install a piece of furniture, you would instead put on a pair of goggles that visually shows how to assembly the parts. Or perhaps a more business worthy example where the goggles connect to the car computer, and the on-board diagnostic (OBD) service tells what is wrong with it. The goggles would then be able to visually show what part to replace and where it is located on the car. Even though there are differences between 4.0 and 5.0, it is doubtful that technologies emerging so close would be given different industrial “revolutions”, which is why both the connectivity and the human-machine integration are often termed as Industry 4.0.

3.5 Industrial Internet Consortium (IIC)

The Industrial Internet Consortium, or IIC for short, is a consortium created by AT&T, Cisco, General Electric, Intel and IBM [32], with more than 200 members [33]. The IICs main focus is to collaborate and share knowledge about emerging technologies and recommendations regarding the Industrial IoT. The IIC does not participate in directly developing open standards for industrial use, but instead focus on designing frameworks, test beds and use cases in areas such as health-care, manufacturing and security [34]. One of the frameworks that has been created by the IIC is the Industrial Internet Reference Architecture, or IIRA for short [35], which is a framework designed to aid in documentation, development and communication of Industrial IoT.

3.6 Industrial Internet Reference Architecture (IIRA)

The IIRA is a reference architecture created by the IIC [35], which was designed to aid in the development of a industrial system with IIoT in mind. The Reference Architecture document is divided in two main parts, where the first is regarding the fundamentals and context for a system under construction but also on different viewpoints. The second part is concerning sections that are under continuous analysis.

The different viewpoints focused on in the Reference documentation are: Business, Usage, Functional and Implementation.

- Business: Focus on the business visions and information that is interesting for business related decision makers.
- Usage: The central systems capabilities and use cases.
- Functional: The devices and components functional relationship between each other and with external parts.
- Implementation: The implementation part of the functional devices and life-cycle plan.

3.7 Cloud security concerning outside threats

There are three main categories for how to secure an industrial environment, the first and easiest to implement is to have no traffic in or out of the enterprise. Only the internal network can be used, making it impossible for external attacks to occur without direct access to the network. However, it also cripples any remote diagnostics or repair of the system that is not performed on site, and makes it impossible to connect it to a cloud service. Another alternative is to only allow traffic going out of the factory and none going in, which will make it possible for an outsider to get continuous data about what is going on inside the factory but not possible to request information, other than what is transmitted. However if the factory transmits all valuable information about sensors and devices to a server located outside of the factory, other computers could then connect to this server and see information about their devices. This would move the security risk out from the factory onto the server and in the case of an attack it would only be the information that is threatened and not the actual control systems, i.e. the Operational Technology (OT) threat has been moved to an Informational Technology (IT) threat. A third alternative is to have all connections preset beforehand, and any deviating or new connections would then simply be denied access. This final method still leaves vulnerabilities against man-in-the-middle and similar attacks.

3.8 Industrial IoT platforms

Platforms concerning IoT are commonly denoted as middleware, as the platform is the distributor of information between the “things”/devices and the more abstract application layer. For a platform to be useful and practical it is necessary that the application scope is large enough to cover the uses and can scale easily with an increasing number of nodes. A platform that is specified and focused to do certain tasks might be overtaken by another, more sluggish implementation if this is more dynamic or has a larger set of possible implementations. However, it might also be that the area of use is so niched that there is really only one platform available that satisfies the demands. The following subsections cover some of the more recognized platforms concerning IoT, some of them with more or less focus on Industrial side of IoT.

3.8.1 Kaa

Kaa is an open-source and free IoT platform [36], which claims to have essential middleware for Industrial IoT application, by having a set of methods that can be used to predict future failures and simplify troubleshooting errors. It is also possible to implement a cluster of Kaa servers located in different geographical locations, and when the primary server either gets disconnected or crashes, a new node is given the “primary” tag. Kaa clients can be executed on a number of systems, from large fast ones to smaller devices with less than 10 KB of RAM [37]. The server or middleware functionality can be implemented on either private servers or by directly using Amazon Web Services (AWS) [38].

3.8.2 Predix

Predix is a proprietary cloud platform [39], created by General Electric (GE) [40] as a Platform as a Service (PaaS), designed for the Industrial IoT field. The platform is sometimes referred to as an “ecosystem of services” or as the “app store for Industrial IoT”, due to the fact that it hosts solutions for a number of different applications. The implementations for the different applications range from anomaly detections, text analytics and machine learning to data management, security services and indoor positioning [41].

3.8.3 OpenRemote

OpenRemote is an open-source project to make it possible for devices, with different protocols, to communicate and cooperate with each other [42]. It initially started as a tool to use in home automation but has now expanded to other applications such as health care and entertainment. The three main features of OpenRemote are integration of devices, user interface design and service management. The managing services are account management, which allows for creation and management of several users, a messaging service for status notifications and data analytics which is used to analyze the received data.

3.8.4 Echelon IzoT

IzoT is a platform family of chips, stacks, interfaces, software development kits (SDK) and management software focused on the Industrial IoT [43]. The name IzoT has no directly abbreviated meaning, however the 'z' can be interpreted as a power of 2, hence the name IzoT could be written as I^2oT hinting at Industrial Internet of Things. Echelon sells IzoT enabled devices and services such as sensors, routers and cloud servers. The devices are multi-protocol enabled and can, amongst other protocols, communicate on the conventional Internet Protocol (IP).

3.8.5 CloudPlugs

CloudPlugs is an IoT cloud service without focus on industrial use [44]. It aids in the development of IoT applications by providing libraries that simplifies the connection to either their cloud service or an in-house server. The primary communication with the cloud service is done using either MQTT or HTTP. See section 3.9 for more information about MQTT, HTTP and other communication protocols.

3.8.6 mnubo

Mnubo's primary goal is to help retrieve the valuable information in a large ocean of sensor data by using smart analytics [45]. The five main focus areas are consumer, smart home, commercial, agriculture and light industrial devices. Light industrial refers to industrial devices that are more focused on the consumer end-use rather than a business, such as heating, ventilation and air conditioning. Other than offering "analytics as a service", by aiding to visualise and analyse data, mnubo has two other services, "data as a service" and "intelligence as a service". "Data as a service" helps to organize, modelize and store data , whereas "intelligence as a service" aids in building predictive and machine learning models.

3.8.7 Nimbits

Nimbits is an open-source IoT Application Programming Interface (API) [46] that is used to process and store timeseries data. The source code consists of a server and a client software that runs on Java embedded devices. The information gathered is structured in a treelike fashion and the server can be set up to either store the data and visually represent it, or to trigger an event to happen when a certain condition is met.

3.8.8 AllJoyn Framework

AllJoyn is an open-source framework to simplify discovery and communication between devices [47]. A combination of mDNS and UDP packets are used to enable communications with IP traffic [48]. By using mDNS it is not necessary to have a dedicated DNS server for "name-to-IP-adress" translation. The fact that it is not necessary to have a dedicated DNS server is important as AllJoyn focuses on IoT connectivity without a cloud service. However, if a cloud service is desired it can also be implemented as an optional added feature.

3.8.9 Node-Red

Node-Red is a tool designed for the IoT by "wiring together hardware devices, APIs and online services" [49]. Node-Red is set up using a web browser by connecting different blocks/nodes together. The blocks/nodes range from publication of web pages, hardware specific functionality such as Raspberry Pi GPIO pin control, sending and receiving MQTT messages to email and twitter messages. Node-Red can be used with a number of systems, such as Raspberry Pi, BeaglehBone, IBM Bluemix and Amazon Web Services.

3.9 IoT Protocols for industry

There are many protocols competing to be the foremost used in different fields of the Internet of Things, some emerging from the IoT directly as completely new protocols and others evolve from already established techniques in industrial applications. A special tailored protocol for IoT might be better in terms of usage and speed, however, that alone does not make it successful. Already established protocols are more acknowledged and accepted and the threshold to move to a new protocol might be steep. Another challenge is that it is necessary that the protocols can be used by essentially all different fields and applications as one of the main objectives in IoT is to connect everything. Instead of, as many previous protocols, being proprietary with closed source code, many of the protocols for IoT are collaboratively being developed by many companies and organizations with public and open-source-code. Having the code developed in "the open" helps both the spread of the protocol, because of easier availability, but also helps in the fact that aid can be received during the development process.

The protocols used can be divided in two main groups which are, client/server and publisher/subscriber. The client/server type relies on point-to-point communication and is therefore often deemed more secure [50]. Hovewer, using point-to-point communication makes it difficult to scale up to large networks.By using a publisher/subscriber type of communication the publisher sends data to a central node, usually called the broker. The broker has connected nodes that subscribe to different topics, and once a publisher has sent new data the broker will forward the data to all subscribing nodes of that topic. See Figures 3.1 and 3.2 for comparison. Some of the predicted protocols to be used in the future for Industrial IoT are OPC UA, HTTP, MQTT, CoAP, DDS and AMQP.

3.9.1 Open Platform Communications Unified Architecture (OPC UA)

OPC UA stands for Open Platform Communications Unified Architecture and is a client/server protocol. The protocol has evolved from OPC, which sometimes

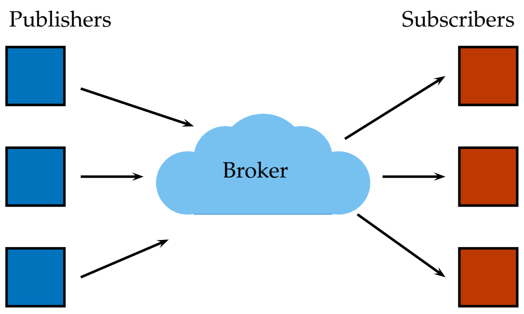


Figure 3.1: Publisher/Subscriber type protocol setup

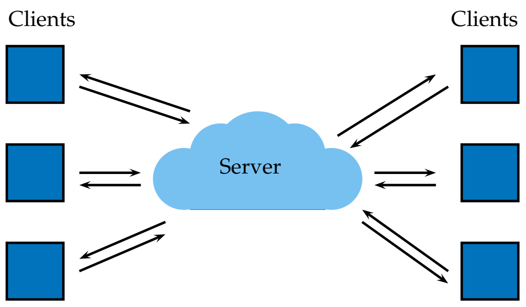


Figure 3.2: Client/Server type protocol setup

is described as “Classic OPC” [50] to help differentiate between the new and old protocol, but at the same time maintain the recognizable name. The new protocol is open-source, but not free, and is used in industrial environments to model and deliver information [51].

3.9.2 Hypertext Transfer Protocol (HTTP)

Hypertext Transfer Protocol (HTTP) [52, 53] is a protocol used to carry hypertext messages and is the main protocol used to transfer data over the Internet. HTTP itself is not very useful as a protocol to use with IoT nodes as it has a large overhead. The data traversing the IoT network usually consist of short messages that update quite frequently, and by having a large overhead on all of these messages congests the network. Another approach to use HTTP, in IoT applications, would be to buffer up some data before sending it, leading to a lower relative overhead. HTTP could therefore be used to send data gathered inside a factory or organization and once every hour or day send it to a cloud service.

3.9.3 Message Queue Telemetry Transport (MQTT)

MQTT stands for Message Queue Telemetry Transport [54, 55] and is a centralized publisher/subscriber type protocol, with a broker that relays messages. Contrary to HTTP, MQTT has a small overhead of about 2 byte [56] and is data agnostic, meaning that the transmitted data does not have to be formatted in a specific way.

3.9.4 Constrained Application Protocol (CoAP)

CoAP stands for Constrained Application Protocol [57] and is a similar protocol to HTTP but made to be lighter with a smaller overhead. CoAP is useful in implementations where data is sent often with small payloads and where HTTP would otherwise be used if not for the large overhead.

3.9.5 Data Distribution Service (DDS)

DDS stands for Data Distribution Service [58] and is a publisher/subscriber type protocol. DDS systems are distributed which means that instead of communicating with a central part, as with MQTT and the broker, all nodes can communicate with each other directly [50]. See Figures 3.3 and 3.4 for comparison of a distributed and centralized type protocol.

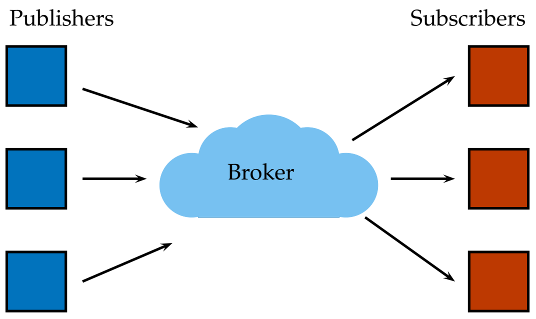


Figure 3.3: Centralised Publisher/-Subscriber type protocol setup as used in MQTT

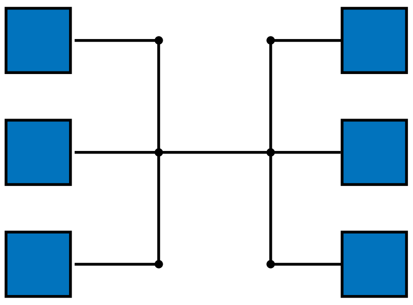


Figure 3.4: DDS distributed system where all nodes are connected to each other

3.10 Security with SSL/TLS encryption

Secure Sockets Layer (SSL) [59] is an obsolete version of the newer encryption protocol Transport Layer Security (TSL) [25, 24]. However, the standard is often referred to as SSL/TLS even when it is the newer TLS that is used. The security is initialized with a series of handshakes between a client and a server.

The client initializes the communication with a “hello” message to the server, which includes what type of acceptable protocols that the client allows. The server then responds with a server “hello” that acknowledges the message from the client, and selects what encryption to use from the allowed list. The server also sends its digital certification. The client then checks that the certification from the server is correct and uses that to encrypt a private key and sends that to the server. Both sides now have a symmetric private key that makes it possible to encrypt the communication between the server and client.

3.11 Redis

Redis is an open-source data-structure database and message broker [60], that supports a variety of different data types such as strings, hashes, lists, sets and bitmaps. A strong feature of Redis is its versatility, as it can can be used directly via a Bash command line or by using libraries for several different languages such as C, C++, C#, Erlang, Java, Node.js, Rust, Go among others. Redis is written and developed in C to be executed on UNIX type operating systems, such as GNU/Linux and OS X, there is currently no support for Windows machines.

4
Network theory

This chapter covers basic information about computer networks which might seem unnecessarily simple to a reader with an engineering background in computer science. However, it is useful to refresh some information about how conventional network models are constructed, to be able to compare them to the industrial variants described throughout the report.

A network is a structure of interconnected computers or nodes, which communicates and shares information with each other. The network can be a local one that only houses a few computers or it can be a global one with millions of connected devices spanning several countries. To be able to communicate between each other, on even the simplest of networks, it is necessary that all involved parties follow a common standard, one such is the Open Systems Interconnection Model [61, 62], or OSI model for short.

4.1 OSI model

The OSI model is a layered standard for communication functions in computer networks. The OSI model contains several layers [61, 62], where the layers communicate directly to their respective on the other side of the communication, i.e. layer 2 on the transmitter side communicates with layer 2 on the receiver side, and it does not know or care about what specifications the other layers have. The OSI model usually consists of seven (7) layers which are the Application, Presentation, Session, Transport, Network, Datalink and Physical layer [61, 62]. Another similar model is the TCP/IP Model that only uses five (5) layers, which is done by merging the three top layers together into one Application layer and also merging together the data link and physical layers [63]. See Figure 4.1 for an illustration of the models compared to each other.

4.2 Industrial networks

The demands on industrial networks are greater than that of commercial products. They have to be more ruggedly designed in terms of physical layout to be able to withstand a harsher environment. Industrial networks also have to be able to provide reliable, consistent information and deterministic real-time services to the end users. An often occurring term in these types of contexts is Industrial Ethernet, which is simply Ethernet used in industrial environments. It is the same as the conventional

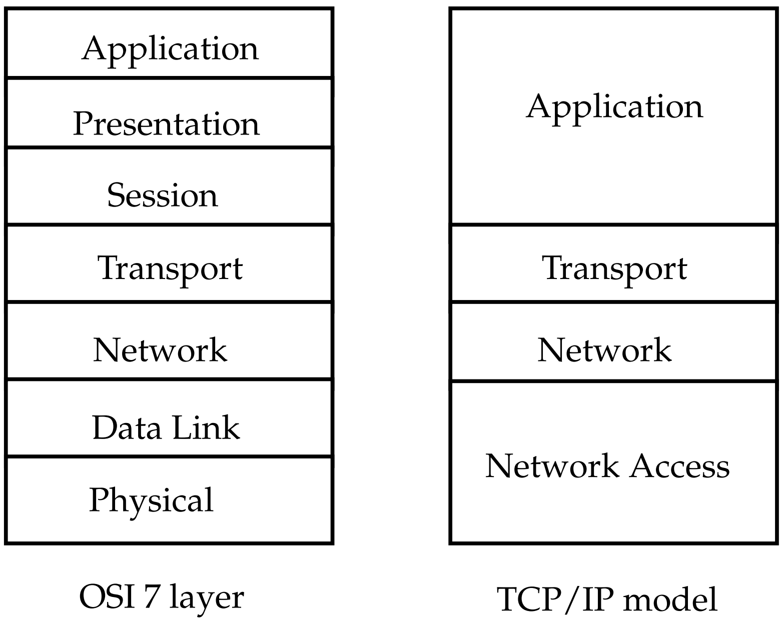


Figure 4.1: OSI 7 layer to the left compared to the TCP/IP model to the right

Ethernet but with more rugged ports, connectors and switches. Special network protocols are used in Industrial Ethernet and there are more than a dozen different solutions [64]. Even though only Profinet is used in this project, it is interesting to see how different industrial Ethernet-based networks function to see similarities and differences, the following subsections describe some of the most widely used ones.

4.2.1 EtherCAT

Ethernet for Control Automation Technology, or EtherCAT for short, is a field bus system [65], which consists of master and slaves. The master is the only one allowed to actively send frames, meaning that no slave is allowed to send frames without being commanded. This is done to maintain a predictable real-time behavior. The master uses a normal MAC protocol whereas the slaves use a special controller. The controller is called the EtherCAT Slave Controller or ESC for short, which processes the message purely in hardware to maintain the real-time service. The messages are sent in standard Ethernet frames and any resulting jitter is less than 1 μs [65].

4.2.2 Ethernet Powerlink

Ethernet Powerlink is a completely software-based real-time field bus system [66]. The implementation is based on standard Ethernet and the real-time demand is created by a mix of polling and time slicing. The system is made up of a Manager Node (MN), and several Controlled Nodes (CN), the Manager Node controls who gets to speak on the network by polling the Controlled Nodes for information. The polling is done in such a way that each CN gets a specific slot in time to speak. A major advantage of Ethernet Powerlink is that it is completely software based which means that it can more easily be implemented on present systems, without any change in hardware, there are also open source protocol stacks such as openPOWERLINK [67] that can be implemented free of charge.

4.2.3 Profibus

Profibus, acronym for Process Field Bus [68, 69], is not based on Ethernet but instead on the serial line RS485 [70]. Profibus is maintained by “Profibus & Profinet International”, the network consists of Master and Slave nodes, where the masters poll the slave devices for controlling and information gathering. The Profibus network has a relative slow transmission speed, compared to the other protocols evaluated above, of about 12 Mbits/s.

4.2.4 Profinet

Process Field Net or Profinet is an industrial Ethernet protocol which like Profibus, is maintained by “Profibus & Profinet International” [71]. Profinet is an open technology, that is independent of vendor. However, to receive information and support it is necessary to be a member of “Profibus & Profinet International”. There are two versions of Profinet; Profinet IO and Profinet CBA [72]. Profinet CBA is the older one [73] and stands for Component Based Automation [74, 72], it is suitable for communication machine-to-machine [73] and is centered around distributing automation applications [75]. Profinet IO, which stands for Input Output, on the other hand is centered around distributing the I/O units. In this project only Profinet IO will be used and no further evaluation or information of CBA will be given.

There are three levels or classes in the IO protocol, Class A, B and C [76]. Normal TCP/IP traffic in Class A, Real-time or RT in Class B and Isochronous Real-time or IRT in Class C. TCP/IP is used for normal low priority traffic and is sent the same way as in conventional TCP traffic, see Section 4.1. The second level, Real-time (RT), can also be used with conventional Ethernet devices and infrastructure. To be able to provide the real-time property, the RT-frames are marked with a VLAN tag with the ID 0 and a priority of 6. These frames are called priority tagged frames [77] and will be given priority over other messages. As of now, no physical difference have been made to implement Profinet, however in the third level, Isochronous Real-time (IRT), special hardware is needed to fulfill the requirements. In Isochronous Real-time the cycle times of transmitted data is less than 1 ms [71]. To solve the problem of having Real-time and at the same time being able to connect to a normal channel, the channel is divided in “time sections”, one channel for IRT specific transmissions and the other an open channel for normal communication. See Figure 4.2 for illustration.

Since IRT traffic does not coexist in the same time frame as the normal traffic there is no need to use the VLAN tag in the transmitted IRT frame. In Figure 4.3 the layers of the different implementations can be seen.

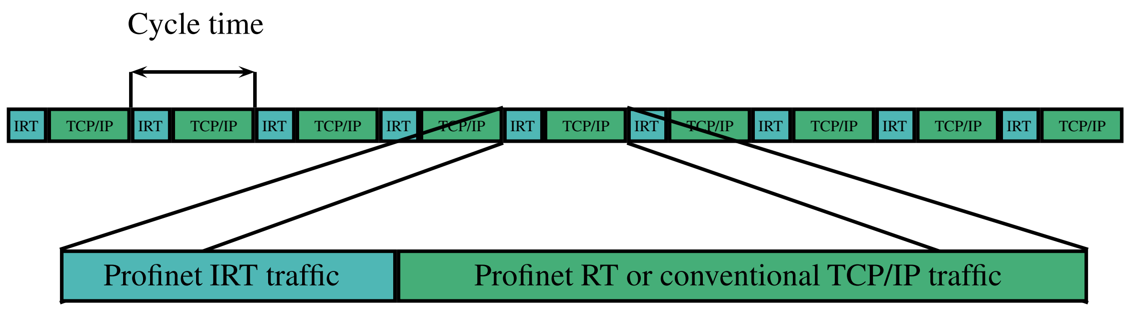


Figure 4.2: How the time is divided to provide both Isochronous Realtime, Realtime and conventional TCP/IP traffic

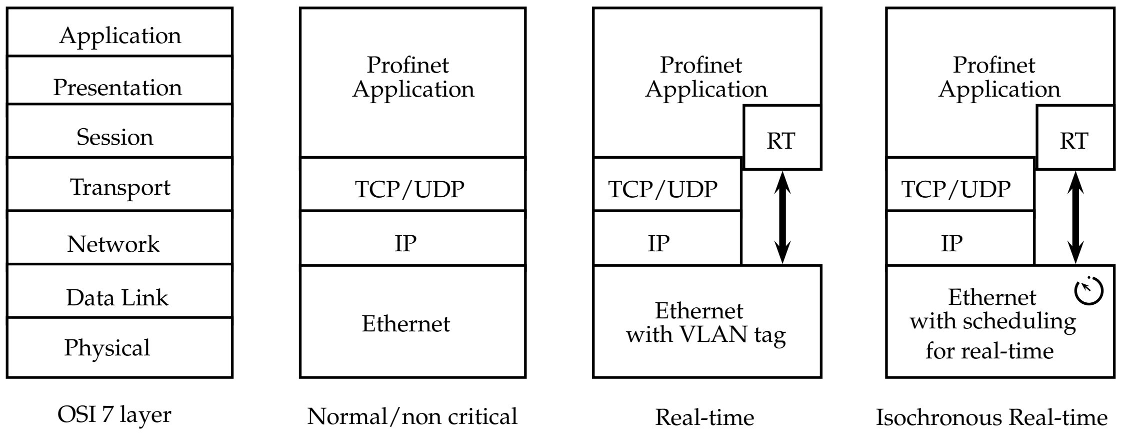


Figure 4.3: OSI 7 layer to the left compared to the different Profinet traffic types

The Profinet IO system contains three different types of devices, IO Controller, IO Device and the IO supervisor [78, 72]. The IO Controller is a PLC or computer that controls and monitors the IO devices. The IO Devices are the nodes that read IO data with sensors or control actuators. The IO Supervisor is used to set different parameters in the IO Controllers but also to diagnose separate IO Devices and the network itself. The IO Devices are configured with General Station Description files or GSD for short.

General Station Description (GSD)

The GSD files are saved in the file format GSDML which is formatted as an XML file. The files contain information about the capabilities of the devices, such as vendor and model name, identification number, maximum polling frequency, supported baud-rates and the configuration of any IO ports.

4.2.5 Intelligent Distributed Automation (IDA)

The basic setup of automation in an industrial environment is that the PLC has control over the I/O nodes directly and micromanages their functions. An illustration of this can be seen in Figure 4.4. This compute-centralised way has some drawbacks in that if one of the I/O nodes needs to be replaced, it is very time consuming as the code in the PLC Controller has to be modified to fit the new node. If this is a large system there can be several thousands of lines of code, and the programmer needs to separate out the ones concerning the replacement node.

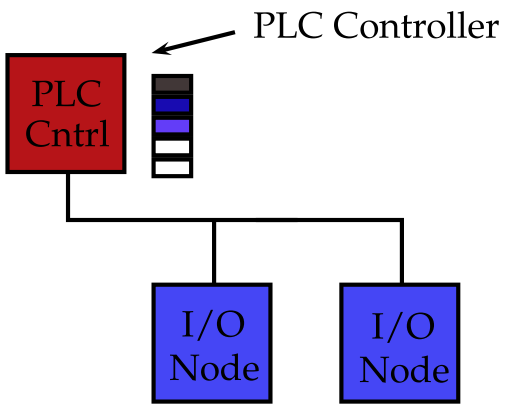


Figure 4.4: Basic setup of PLC network, where all instructions are commanded by the controller

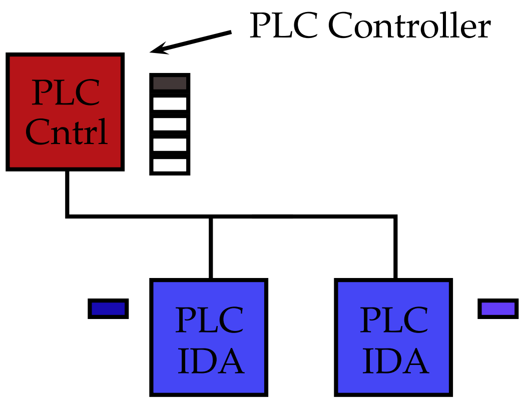


Figure 4.5: IDA setup of PLC network, where the code is distributed to the I/O devices

The company SEVAB has developed a technique to simplify the transition when implementing new nodes. Instead of having the PLC manage each and every I/O node, the code is instead distributed across what they call PLC IDA nodes. Each IDA node has its own code to control the functionality, and the PLC Controller instead has more simple instructions for the nodes, such as start and stop. When implementing new nodes, the code can remain the same in the controller since the new node can have the same abstract functions as the last one, i.e. start or stop. See Figure 4.5 for illustration.

4.3 Wireshark

Wireshark is an open-source network protocol analyzer [79], that can be used under several platforms such as GNU/Linux, Windows and OS X. Wireshark can be used to inspect network traffic, troubleshoot firewalls and decode several encapsulations and protocols, one being Profinet [80].

4.4 Personal Area Network (PAN)

Personal Area Network, PAN for short, is a close range, usually around 10 meters [81], wireless network that is centered around one person [82]. There are a number of different technologies, but only Bluetooth will be used in the thesis project so no other techniques will be compared.

Bluetooth is a standard for wireless communications that was developed by the Swedish company Ericsson in 1998 [83]. Unlike other PAN network technologies in the nineties, Bluetooth does not transmit infrared waves, which has a frequency range between 430 THz and 300 GHz, but is instead located around 2.4 GHz [84]. The current version of Bluetooth is 5.0, which has a transfer speed of around 2 MBit/s. A frequently used term in Bluetooth contexts is “Bluetooth Low Energy”, also known as Bluetooth LE or BLE [85]. BLE is a sub-part of Bluetooth 4.0 which has a different protocol than that used in conventional Bluetooth to reduce the energy consumption.

5
Gateway design

The designed demonstrator/prototype is a gateway that can be connected to a Profinet network on one side and to an arbitrary network or technology on the other, such as Bluetooth or WiFi. It is also possible to connect the gateway as a "proxy", separating two Profinet networks from each other, see section 5.3.2. The following sections covers the approach and layout of the gateway, the overview of the gateway can be seen in Figure 5.1 and each sub-part is discussed in their representative section. The first sections cover some basic background regarding where the information was collected and what underlying software and hardware was used.

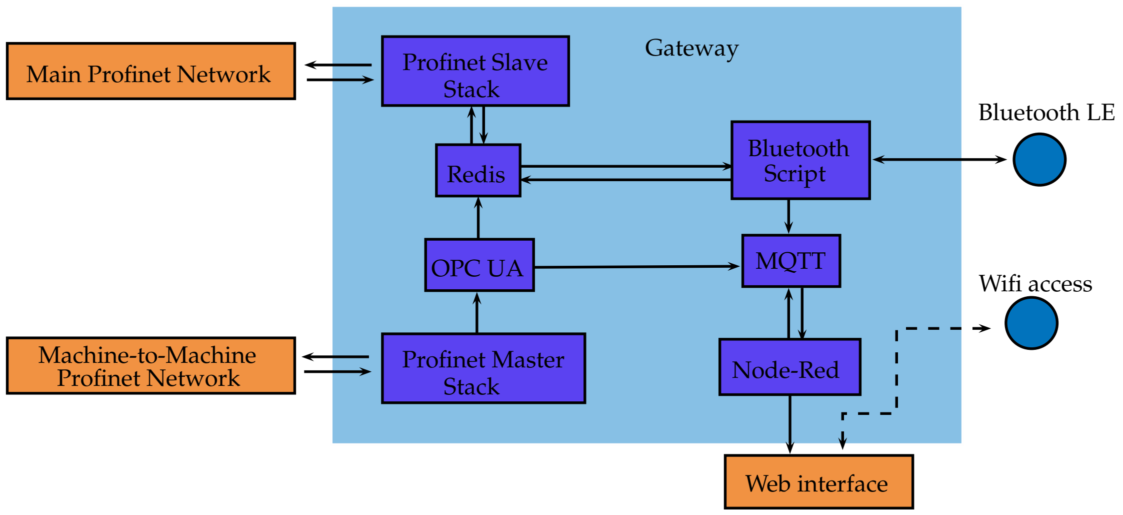


Figure 5.1: Overview of how the Gateway is structured internally

5.1 Profinet

To gather the necessary information and knowledge about how the Profinet protocol is implemented, the automation community Profibus and Profinet international (PI) was used [86]. Usually a yearly fee is paid to remain a member, however, by carrying out non-profit research, a free license was used during the 20 weeks of the thesis.

The reason to why Profinet was used and not any of the other Industrial Ethernet protocols is that Siemens is one of the largest suppliers of PLCs in Europe and they use Profinet, but also because the demonstrator should be able to fix the addressing problem in section 1.1.

5.2 Hardware and software version

Some time was spent on what type of hardware could be used that both runs on GNU/Linux and is dependable enough to be used in an industrial environment. However, even though there exists hardware that suits both of these demands, a Raspberry Pi 3 was used in the project to simplify the implementation and to reduce the time necessary before prototyping could be done. The development was done on “Raspbian Jessie with Pixel” [87], kernel version 4.4.

Even though the Raspberry Pi 3 does not comply with the industrial demands on hardware, there is a “Raspberry Pi 3 Compute Module” that is used in already available industrial Profinet gateways such as in the “Kunbus Profinet IRT Gateway” [88]. Since it is the same microprocessor in the Compute Module as in the Raspberry Pi 3, porting the software to the compute module should be trivial. Note that the Kunbus Gateway only acts as a slave and not as both controller (master) and device (slave) as the designed prototype/demonstrator does.

5.3 Profinet stack implementation

The initial approach was to evaluate whether it would be feasible to create a stack from the core. However, after some investigation by reading the documentations from PI and analyzing the network traffic with Wireshark, it is a large undertaking to get all functionality to work correctly but also to be able to certify any final product. By implementing an existing software stack this certification time is much shorter. The problem has therefore been moved from creating a stack from scratch to finding already available master and slave stacks that can be merged together.

5.3.1 Profinet slave stack

The slave stack was received from the German company “Port” [89], and comes as source code written in the programming language “C”. The slave software is compiled together with the hiredis library and related libraries. The Profinet slave stack on the gateway communicates with the main Profinet network and therefore with the original Profinet Controller (master), this controller has to be set up which is done using SIMATIC 7. In Figure 5.2 the layout is illustrated, however as earlier described, in section 1.1 and Figure 1.3, from the controller point of view the secondary Profinet network is unknown and only the gateways’ interface has to be considered. The setup in SIMANTIC 7 therefore only has the gateways’ Profinet slave stack as a module, see Figure 5.3 for SIMATIC 7 layout.

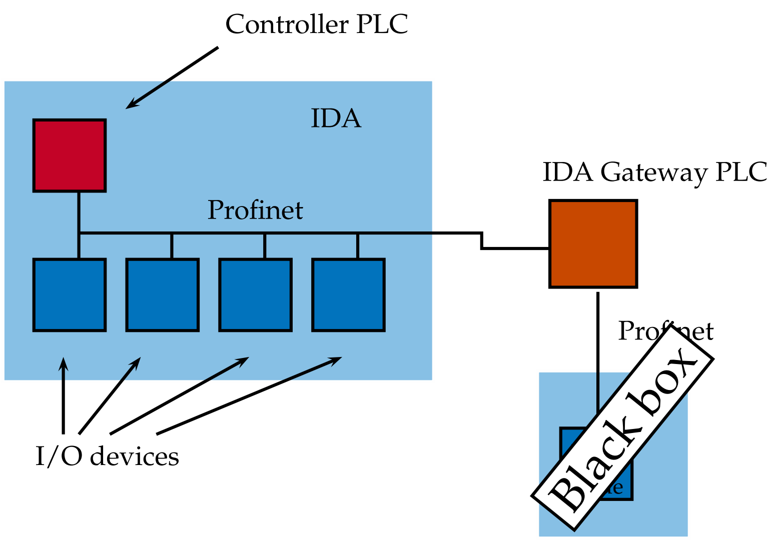


Figure 5.2: Network as seen from the Profinet Controller (master)

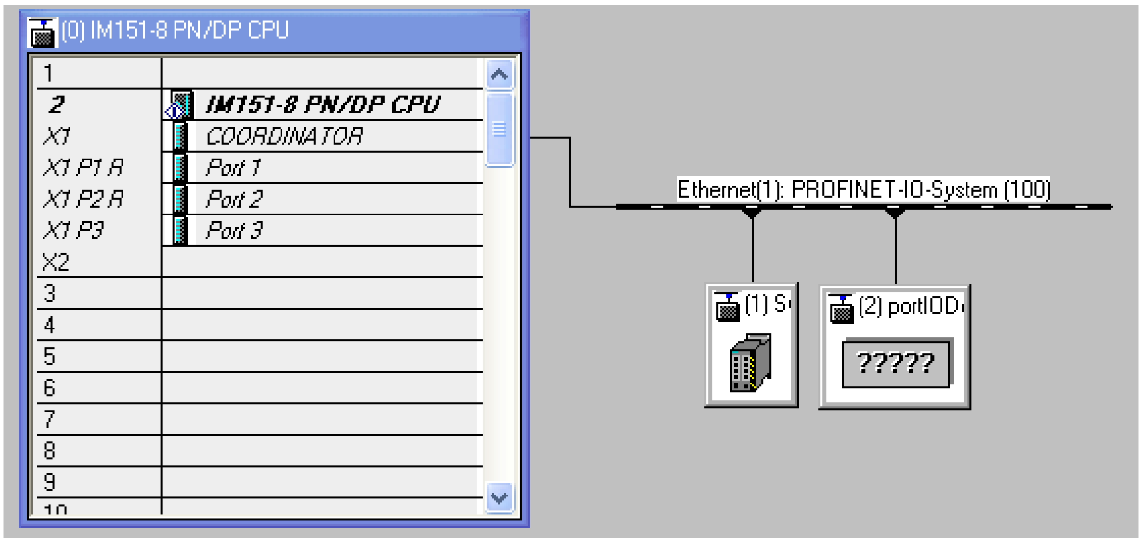


Figure 5.3: Profinet Controller (master) layout for the modules. The large window to the left is the controller itself, in the middle a Profinet switch and to the right is the gateway

5.3.2 Profinet master stack

The master stack is used to set up the gateway to work as a Profinet Controller that controls one I/O device or Profinet PLC. To be able to have both master and slave properties on the gateway it is necessary to have two separate networks and therefore two physical Ethernet ports on the gateway.
As the Raspberry Pi 3 is used as the gateway demonstrator, which only has one (1) Ethernet port, it is necessary to add another port. This port is added using a USB-to-Ethernet adapter.

The master stack used comes from “Codesys” [90], and is set up by first installing a Debian package on the Raspberry Pi and then, via their development system, set up any slave devices. Once starting the program it will use network port 34964 on all available addresses on the local machine (0.0.0.0:34964), making it impossible for any other application to communicate on the Profinet network. A fix for this can be found in Appendix E. The setup is similar to that of SIMATIC 7 and can be seen in Figure 5.4.

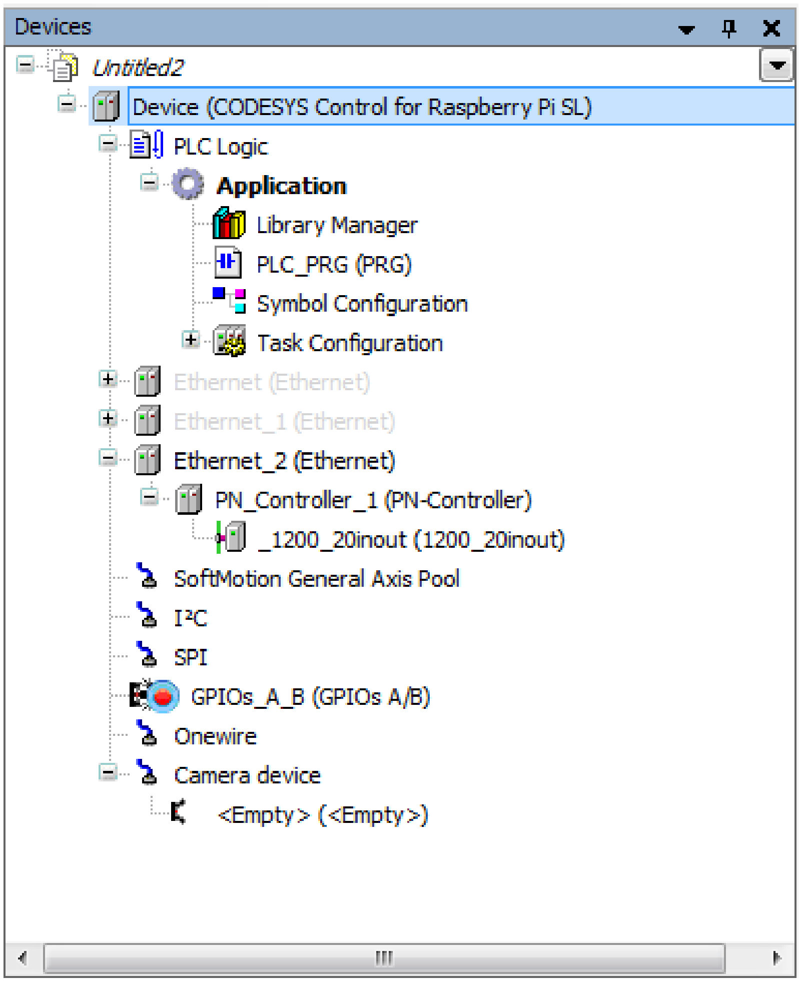


Figure 5.4: Setup in Codesys, where 1200_20inout is GSD file for the Profinet I/O device used

5.4 Gateway master communication with slave

As the master stack is implemented with the program Codesys, and not as C source code, it is not possible to add the hiredis C library. However, it is possible to implement OPC-UA server in Codesys that transmits the desired signals. Another program then listens to the messages transmitted from Codesys and relays them to both the Redis database and to Node-Red. The idea behind sending the data to both Redis and Node-Red is to make the program easier to implement with another

Profinet Controller (master) stack. If another stack were to be implemented it could send the data to Redis and be used by any other program that wants the information. By using MQTT to send the traffic to Node-Red it is easy to re-route it to another web server.

5.5 Network traffic analysis

In Figure 5.5 the normal network layout is shown. There are a few different ways to analyze the traffic between the Controller and the I/O devices (PLC IDA). The easiest way would be to implement a hub between the desired I/O device and the Profinet switch. A hub works by forwarding all incoming messages to all other ports, making it possible to sniff the traffic by just connecting to one of the other ports. By connecting a computer with Wireshark installed to *3 in Figure 5.6 it is possible to see the traffic between the Controller and the Device. The biggest problem with this approach, however, is that hubs are rarely used today and are therefore somewhat difficult to obtain.

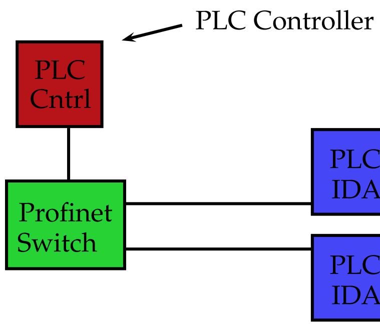


Figure 5.5: Normal network setup without network analysing

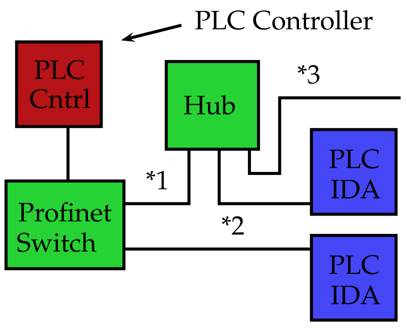


Figure 5.6: Network setup with hub to extract network messages

If we were to implement a switch instead of the hub, it would only forward the messages from the Controller to the IO Device, i.e. from *1 to *2 in Figure 5.7. It is possible to configure some switches to act like a hub and forward the traffic to port *3 as well. In the project a TP-link TL-SG1005E switch was used to forward the traffic however, only non real-time traffic was forwarded to port *3 and the high priority real-time messages were only sent to port *2. A dirty hack to fix this would be to flood the switch with fake MAC addresses, which would fill up the memory and make it impossible for the switch to know which port the IO device is located, forcing it to send the data to all ports.

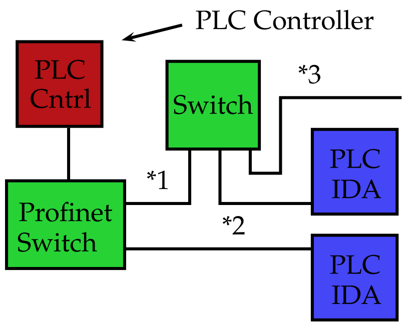


Figure 5.7: Network setup with a switch used to extract network messages between PLC Cntrl and the top PLC IDA

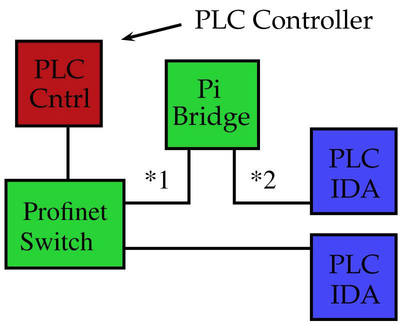


Figure 5.8: Network setup with a Raspberry Pi configured as a network bridge that can analyse the traffic

Instead of filling the memory of the switch with fake adressess, another Raspberry Pi was placed in the position of the switch and configured to act as a bridge between the two networks. The commands necessary to set up the bridge can be found in Appendix D. By implementing the bridge as can be seen in Figure 5.8, Wireshark can be executed on the Raspberry Pi analyzing any of the ports (*1 or *2) to show the real-time network traffic. The initial use of the Pi Bridge was to sniff the traffic and to try to replicate it with another program, however this proved to be too time consuming and the bridge was instead used to verify that correct packets were sent and that there were not to many lost packets.

5.6 Redis

As one of the goals in the project was to create a demonstrator that could be used as a platform in the future, it is useful that the program is flexible and dynamic. It is also important that the device can be extended with more functions in the future. By using Redis it is easy to implement additional features, as the database structure can be accessed with a number of libraries for different languages. If additional functionality is already implemented in in another system, porting it to fit could be as easy as to implement the Redis library and send any relevant data to it.

In the project Redis clients for “Bash” and “C” were used. Before starting the clients it is necessary to have the “Redis Server” installed and running on your machine, see Appendix B for setup.

5.7 Bluetooth alarm and signal strength

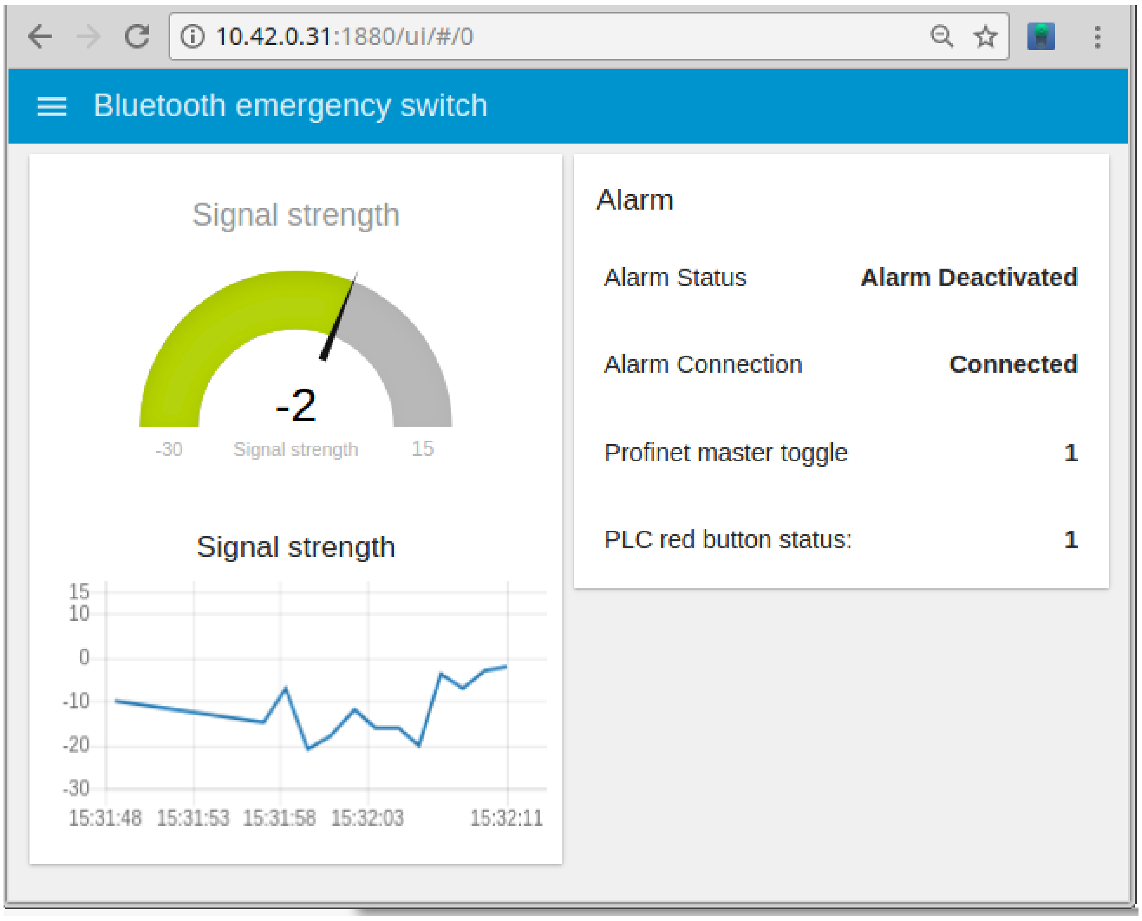
To simulate an emergency stop that can give a rough estimate of distance, a Bluetooth Remote Shutter from Linocell was used. The remote has two buttons, one being used to activate and the other to deactivate the alarm. The distance to the emergency stop is measured by polling the strength of the Bluetooth signal twice every second (2 Hz). This gives a very poor estimate of the distance since just placing an object between the transmitter and receiver will decrease the signal strength drastically. However, since the objective is to demonstrate the distance measuring functionality rather than an actual application, the distance accuracy is not crucial.

5.8 Message Queue Telemetry Transport (MQTT)

To be able to send the signal strength, alarm signals and other messages to a remote cloud server or similar IoT device, the protocol MQTT was used. However, instead of sending the messages to an outside server the Gateway works as both publisher, broker and subscriber. Publications are made in the program that handles the Bluetooth communication with the emergency alarm, and sent to the local broker that forwards the messages to the Node-Red web-client. See Appendix C for on how to set up MQTT.

5.9 Node-Red

Node-Red was used because it is one of the simplest and fastest ways to implement a web interface with aid for MQTT and for a graphical interface. Node-Red was set up to show a graph and a bar of the current Bluetooth signal strength and with four text values, the status of the alarm, the connectivity of the alarm, the toggling value from the Profinet Controller stack and the status of the red button, see Figure 5.9 for layout.



Alarm Status	Alarm Deactivated
Alarm Connection	Connected
Profinet master toggle	1
PLC red button status:	1

Figure 5.9: Node-Red web interface as seen from Chromium Browser

Raspbian Jessie was used during the project which, as of November 2015, has Node-Red preinstalled. However as it does not come with the most updated version it is necessary to run an update.

5.10 Gateway access-point function

For easy access without an Ethernet connection, the gateway is set up to work as a WiFi access point. It is not broadcasting as a public device, but instead it is necessary to have the SSID to access it. By implementing this wireless function on the Gateway it is possible to gain access with a large variety of devices such as laptops, tablets and phones. Figures 5.10 and 5.11 illustrates what the interface looks like from a phone.

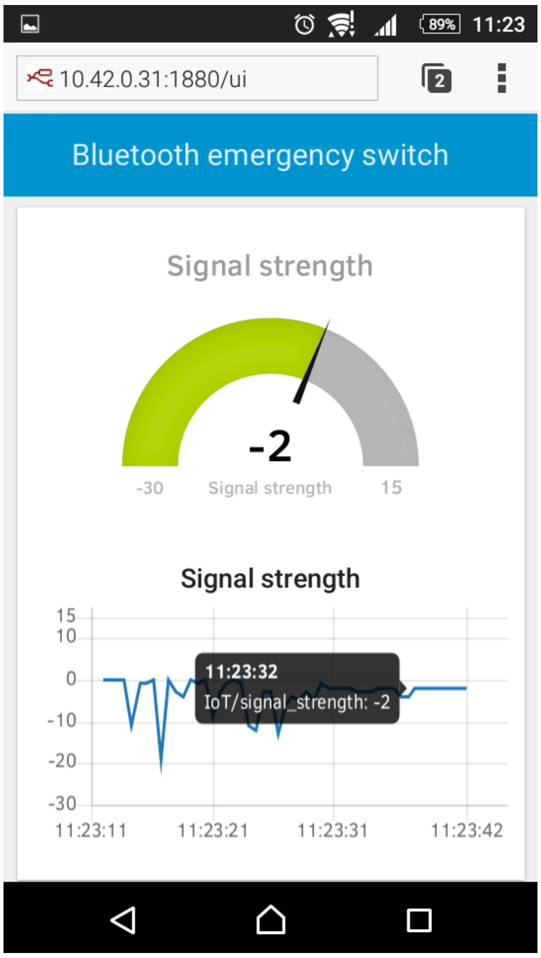
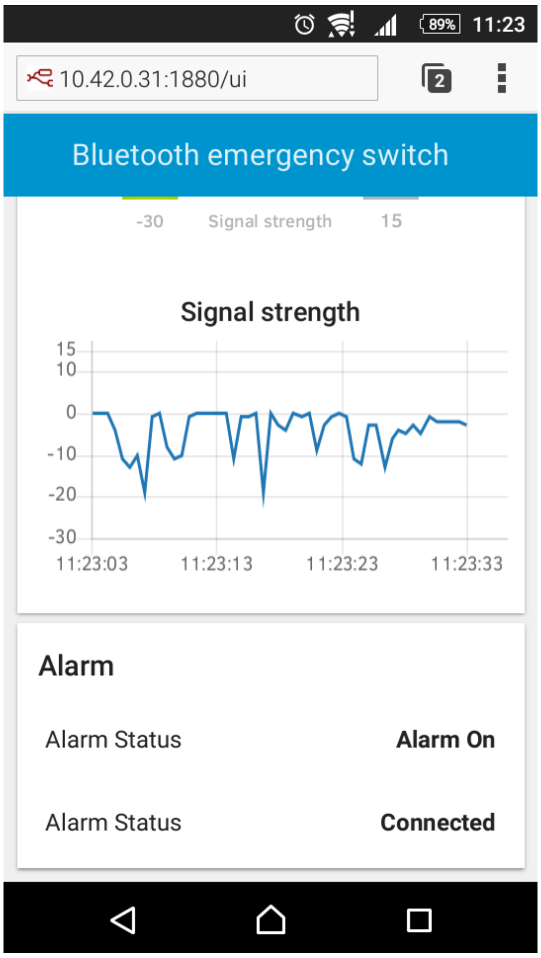


Figure 5.10: Webservice as seen by a phone, with a graph and dashboard of signal strength.



Alarm

Alarm Status	Alarm On
Alarm Status	Connected

Figure 5.11: Lower part of the web-service, where the signal strength graph and alarm status can be seen.

5.11 PLC layout and programming

The test system or demo rig consists of six devices; a WiFi router, a PLC controller (master), a Profinet switch, a 24V DC power-supply, an I/O device (slave) and the Gateway. See Figure 5.12 for the actual components.

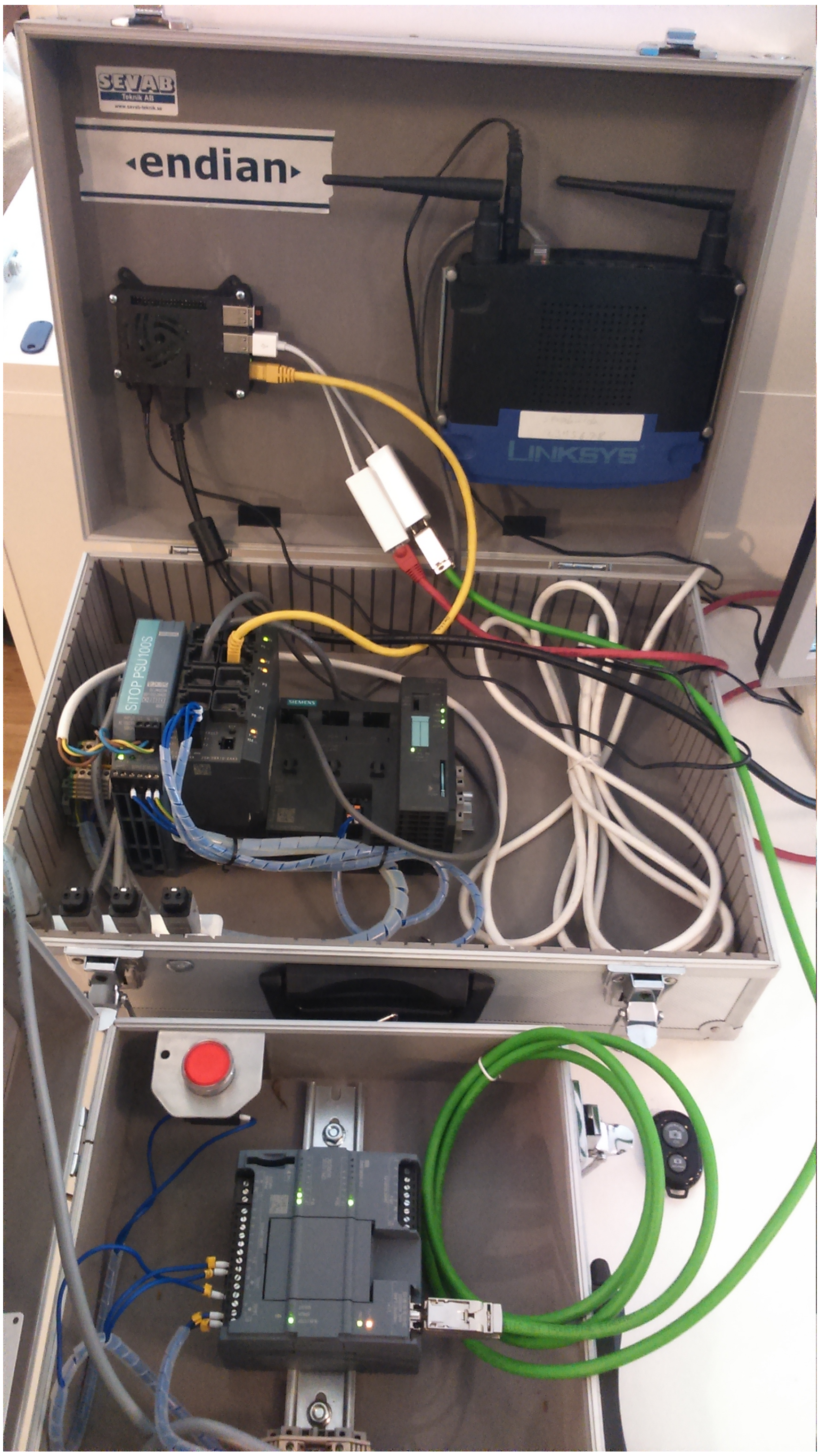


Figure 5.12: Photo of how the actual system hardware is connected. Top left is the Raspberry Pi Gateway. Top right is the wireless router. The section in the middle contains the power-supply to the left, the Profinet switch in the middle and the Profinet Controller/Master to the right. The I/O device (1200_20inout) with the red button can be seen at the bottom.

To prove that it is possible to receive data from the Profinet controller/master as well as from the I/O device, a simple “bit toggle generator” is used. The generator is created as a ladder diagram and can be seen in Figure 5.13. The output from the generator, Q0.0, is sent using OPC UA to node-red where it is displayed in the web browser interface.

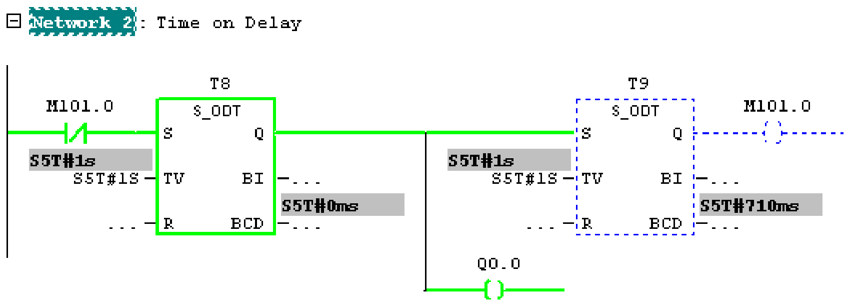


Figure 5.13: Bit toggle generator: Output, “Q0.0”, oscillates between 0 and 1, holding each value one second each (0.5 Hz cycle)

6
Conclusion

The focus of this thesis was to evaluate what protocols and platforms are and will be used for IIoT, and also developing a demonstrator that shows a solution to an actual IIoT problem. The cornerstone of the problem lies in the difficulty of updating and replacing nodes in already established systems. The created demonstrator provides a way to simplify the implementation of new nodes by acting as an abstraction layer between the system and the new device. The demonstrator or gateway also provides a way to get information from the factory floor to an outside network.

The results of this thesis project are a theoretical part with an overview of how and what the Industrial Internet of Things is and on what future trends are to be expected, the project also resulted in a practical demonstrator gateway used to show how to help in the problem of adding or changing new nodes in an already established system.

The verification of the demonstrator gateway was performed both in separate steps throughout the project but also as a whole. The Bluetooth alarm device was tested separately by connecting it to Node-Red and verifying the functionality, and then tested together with the rest of the system. The Profinet drivers were verified in two ways. The first by checking the led status indication on the PLC Profinet I/O (slave) device and the PLC Profinet Controller (Master), the second way of verification was done as described in section 5.5, by using Wireshark to analyze the traffic going in and out of the demonstrator gateway.

6.1 Goals vs Results

The thesis had four goals to fulfill, which can be found in section 1.2, where two of them were regarding a prestudy on network protocols and security concerns with a gateway connected to the internet. The two practical goals were to first create the functionality to connect to the Profinet network on one side and then on the other side be connected to either a cloud platform or to a wireless sensor on the other. The second practical goal was add an abstraction level between an already established system and a new node. Regarding the first two goals the theoretical information can be found in section 3.8 and 3.7. The created demonstrator both has the gateway functionality to talk to Profinet and to the wireless sensor, and also to act as the proxy between an established system and a new node. One thing that is not quite up to specification is that the gateway should be implemented without any alterations to the established system, which is not fulfilled. The required modification is that one must remove the old PLC node in the PLC Profinet Controller (master) and replace it with the GSD file of the gateway. However, after this initial modification, the node on the far side of the gateway can be replaced without any further modification of the PLC software.

6.2 Critical discussion

One major objection to the approach taken in this project is that we initially believed that the same stack could be used to implement both the master and slave, which is incorrect. Therefore another stack had to be found in the middle of the project, and a lot of time was spent later to obtain the software. With better planning we could have received both stacks simultaneously.

Another objection can be made regarding the master Profinet stack, as it is used as an executable file rather than from actual source code, which would have been preferable.

To be able to implement the gateway in an industrial environment it is not only necessary with more rigorous testing and bug removal, but also to implement the software in a real-time operating system. Additional testing, besides stability and durability, would be to test the gateway with more than one PLC I/O (slave) device, to show that the gateway does have dynamic functionality. The closest thing to testing with another PLC was to verify that the PLC controller (master) works both when there is an active PLC I/O (slave) device and when it is deactivated.

6.3 The role of the technology

The implemented demonstrator gateway has a large dynamic scope of uses, as one of the main priorities was to be able to connect Profinet on one side and an arbitrary technology on the other side. The proxy service function where Profinet is connected on two sides has a more narrow use-case, as its intended use is to make it easier to add PLC devices without having to know what type of hardware is connected on the opposite side. Hopefully the idea of having personal emergency switches inside helmets will be a reality and not just the bulky wireless emergency switches that currently exists.

As for IIoT as a whole the role and use is vast: with connectivity comes more frequent information updates, more predicable maintenance and a clearer overview of systems by analyzing them. The benefits will lead to better economy for the users, but also help create a more climate friendly and sustainable industry.

6.4 Continued development

A natural continuation of the project would be to create an actual industrial application with a more rugged hardware, perhaps using a Raspberry Pi 3 compute module. Also other factors regarding stability with more rigorous testing of the software and hardware and also regarding security and reducing power consumption with lowered processor utilization. To create a commercial product it would be desirable to have both the Profinet stacks as “C” source code from the same vendor to get a better and cheaper implementation.

Another important continuation of the gateway project would be to implement the possibility for more protocols to be used and not only Profinet. It would be difficult to implement protocols that require special hardware. However, protocols that can be implemented purely in software, such as POWERLINK, could be a desired addition.

Bibliography

[1] Gilchrist Alasdair. *Industry 4.0.* Berkeley, CA:Apress, 2016.

[2] Sabina Jeschke. *Industrial Internet of things.* Springer International Publishing, 2017.

[3] *What Is IIoT? The Industrial Internet of Things.* 2016. URL: `https://inductiveautomation.com/what-is-iiot` (visited on Mar. 2017).

[4] *PROFINET supports a cloud-based control system.* 2016. URL: `http://www.profibus.com/technology/case-studies/case-study-detail-view/article/profinet-supports-a-cloud-based-control-system/` (visited on Mar. 2017).

[5] Bogdan Wilamowski and David Irwin. *The industrial electronics handbook.* 1st ed. CRC Press, 2011.

[6] Juergen Jasperneite and Joachim Feld. *PROFINET: An Integration Platform for heterogeneous Industrial Communication Systems.* IEEE Conference on Emerging Technologies and Factory Automation, 2005.

[7] Andrew Parr. *Industrial control handbook.* 1st ed. OXFORD: NN, 1998.

[8] William Bolton. *Programmable logic controllers.* Newnes, 2009.

[9] *Automation för industri, scada, el, robotsystem | SEVAB Teknik.* 2016. URL: `http://www.sevab-teknik.se/nyheter/` (visited on Mar. 2017).

[10] Naresh Gupta. *Inside Bluetooth Low Energy.* Artech House, 2013.

[11] P. KG. *PHOENIX CONTACT | PROFICLOUD Technology.* 2016. URL: `https://www.phoenixcontact.com/online/portal/pc?urile=wcm:path:/pcen/web/offcontext/insite_landing_pages/e7863ba5-449d-4f45-98b3-c9f5cc700145/e7863ba5-449d-4f45-98b3-c9f5cc700145` (visited on Mar. 2017).

[12] *What is free software?* 2016. URL: `https://www.gnu.org/philosophy/free-sw.en.html` (visited on Mar. 2017).

[13] Karl-Heinz John and Michael Tiegelkamp. *IEC 61131-3: Programming Industrial Automation Systems.* Springer, 2001.

[14] *Control IEC 61131-3.* 2017. URL: `http://www.rtaautomation.com/technologies/control-iec-61131-3/` (visited on Mar. 2017).

[15] *Programmable Logic Controllers (PLC) CHapter 6 - Ladder Logic.* 2017. URL: `https://www.allaboutcircuits.com/textbook/digital/chpt-6/programmable-logic-controllers-plc/` (visited on Mar. 2017).

[16] *Why does the UK lag in its use of industrial robots?* 2016. URL: `http://smartmachinesandfactories.com/news/fullstory.php/aid/28/_Why_does_the_UK_lag_in_its_use_of_industrial_robots_.html` (visited on Mar. 2017).

[17] Barty Weil. *How Wireless Can Help Expand E-Stop Safety*. 2016. URL: `http://www.rockwellautomation.com/global/news/the-journal/detail.page?pagetitle=How-Wireless-Can-Help-Expand-E-Stop-Safety&content_type=magazine&docid=27419336548d0a7852a82a10ca293783` (visited on Mar. 2017).

[18] *Enhanced Temperature Device Support*. 2017. URL: `https://www.altera.com/products/common/temperature/ind-temp.html` (visited on Mar. 2017).

[19] Octagon systems. *Mining Vehicle Computers*. 2016. URL: `http://www.octagonsystems.com/industries/mining` (visited on Mar. 2017).

[20] Belden.com. *A Robust Industrial Ethernet Infrastructure*. 2009. URL: `https://www.belden.com/docs/upload/Industrial_Ethernet_WP.pdf` (visited on Mar. 2017).

[21] Dirk Slama et al. *Enterprise IoT: Strategies and Best Practises for Connected Products and Services*. O'Reilly Media, 2015.

[22] Srini Srinivasan. *Cloud Computing basics*. Springer New York, 2014.

[23] Mohammad Aazam and Eui-Nam Huh. "Fog Computing: The Cloud-IoT/IoE Middleware Paradigm". In: *IEEE Potentials* (2016).

[24] Networking 101. *Transport Layer Security (TLS)*. 2013. URL: `https://hpbn.co/transport-layer-security-tls/` (visited on Mar. 2017).

[25] Margaret Rouse. *Transport Layer Security (TLS)*. 2014. URL: `http://searchsecurity.techtarget.com/definition/Transport-Layer-Security-TLS` (visited on Mar. 2017).

[26] *Industrie 4.0*. 2016. URL: `https://industrie4.0.gtai.de/INDUSTRIE40/Navigation/EN/Topics/industrie-4-0.html` (visited on Mar. 2017).

[27] Britannica Academic. "Industrial Revolution". In: *Encyclopaedia Britannica* (2016). http://academic.eb.com/levels/collegiate/article/423703502.toc.

[28] Dimitri Volkmann. *The Rise of Digital Twins*. 2016. URL: `https://www.ge.com/digital/blog/rise-digital-twins` (visited on Mar. 2017).

[29] "Digital Twins for IoT Applications". In: *Oracle Corporation* (Jan. 2017).

[30] Shermine Gotfredsen. *Bringing back the human touch: Industry 5.0 concept creating factories of the future*. 2016. URL: `http://www.manmonthly.com.au/features/bringing-back-the-human-touch-industry-5-0-concept-creating-factories-of-the-future/` (visited on Mar. 2017).

[31] Nassir Navab et al. "Scene Augmentation via the fusion of Industrual Drawings and Uncalibrated Images With A view To Marker-less Calibration". In: *Siemens Corporate Research* (2016).

[32] *Frequently Asked Questions*. 2017. URL: `http://www.iiconsortium.org/faq.htm` (visited on Mar. 2017).

[33] *Search members*. 2017. URL: `http://www.iiconsortium.org/cgi-bin/iicmembersearch.pl` (visited on Mar. 2017).

[34] *Case studies from members*. 2017. URL: `http://www.iiconsortium.org/case-studies/index.htm` (visited on Mar. 2017).

[35] "Industrial Internet Reference Architecture". In: *Industrial Internet CONSORTIUM* 1.7 (2015).

[36] *Kaa IoT Development Platform overview*. 2017. URL: `https://www.kaaproject.org/overview/` (visited on Mar. 2017).

[37] *Key Capabilities of the Kaa IoT platform: Connectivity, integration, and Data Processing*. 2017. URL: https://www.kaaproject.org/platform/#hardware (visited on Mar. 2017).
[38] *Amazon AWS*. 2017. URL: https://aws.amazon.com/ (visited on Mar. 2017).
[39] *Built on Cloud, Foundry, Predix is optimized for secure connectivity and analytics at scale - in the Cloud and on the Edge*. 2017. URL: https://www.predix.io/ (visited on Mar. 2017).
[40] *GE Imagination at work*. 2017. URL: http://www.ge.com/ (visited on Mar. 2017).
[41] *Services and Software*. 2017. URL: https://www.predix.io/catalog/services (visited on Mar. 2017).
[42] *About OpenRemote*. 2017. URL: http://www.openremote.com/about/ (visited on Mar. 2017).
[43] *IzoT Platform*. 2017. URL: http://www.echelon.com/izot-platform (visited on Mar. 2017).
[44] *Connecting Things*. 2017. URL: https://cloudplugs.com/ (visited on Mar. 2017).
[45] *Plans*. 2017. URL: http://mnubo.com/plans/ (visited on Mar. 2017).
[46] *nimbits platform*. 2017. URL: http://bsautner.github.io/com.nimbits/ (visited on Mar. 2017).
[47] *AllJoyn Framework*. 2017. URL: https://allseenalliance.org/framework (visited on Mar. 2017).
[48] *Documentation*. 2017. URL: https://allseenalliance.org/framework/documentation/learn (visited on Mar. 2017).
[49] *Overview*. 2017. URL: https://nodered.org/ (visited on Mar. 2017).
[50] Aron Semle. *IIoT Protocols to Watch*. 2015. URL: http://www.automation.com/library/white-papers/iiot-protocols-to-watch (visited on Mar. 2017).
[51] RTA Automation. *OPC UA Overview*. 2017. URL: http://www.rtaautomation.com/technologies/opcua/ (visited on Mar. 2017).
[52] *HTTP - Hypertext Transfer Protocol*. 2014. URL: https://www.w3.org/Protocols/ (visited on Mar. 2017).
[53] Margaret Rouse. *HTTP (Hypertext Transfer Protocol)*. 2006. URL: http://searchwindevelopment.techtarget.com/definition/HTTP (visited on Mar. 2017).
[54] *MQTT FAQ*. 2017. URL: http://mqtt.org/faq (visited on Mar. 2017).
[55] Chu Bin Tang. *Explore MQTT and the Internet of Things service on IBM Bluemix*. 2015. URL: https://www.ibm.com/developerworks/cloud/library/cl-mqtt-bluemix-iot-node-red-app/ (visited on Mar. 2017).
[56] Rahul Gupta. *5 Things to Know About MQTT - The Protocol for Internet of Things*. 2014. URL: https://www.ibm.com/developerworks/community/blogs/5things/entry/5_things_to_know_about_mqtt_the_protocol_for_internet_of_things?lang=en (visited on Mar. 2017).
[57] Coap Technology. *CoAP*. 2016. URL: http://coap.technology/ (visited on Mar. 2017).

[58] DDS. *What is DDS?* 2016. URL: http://portals.omg.org/dds/what-is-dds-3/ (visited on Mar. 2017).

[59] Holly Lynne MCKinley. *SSL and TLS: A Beginners Guide.* 2003. URL: https://www.sans.org/reading-room/whitepapers/protocols/ssl-tls-beginners-guide-1029 (visited on Mar. 2017).

[60] *Introduction.* 2017. URL: https://redis.io/topics/introduction (visited on Mar. 2017).

[61] Rachelle Miller. "The OSI Model: An Overview". In: *SANS Institute* 1.2e (2001). URL: https://www.sans.org/reading-room/whitepapers/standards/osi-model-overview-543.

[62] Andrew Butterfield. *A dictionary of computer science.* Oxford University Press, 2016.

[63] Ronald Wyllys and Philip Doty. *Notes on the 5-Layer and 7-Layer Models of Interconnection.* 2017. URL: https://www.ischool.utexas.edu/~l38613dw/readings/NotesOnInterconnection.html (visited on Mar. 2017).

[64] Mark Hoske. *Ethernet protocols used for industrial applications.* 2011. URL: http://www.controleng.com/single-article/ethernet-protocols-used-for-industrial-applications-governing-organizations-urls/5ea12eb9b5ac6a4e8a31e57470898e67.html (visited on Mar. 2017).

[65] *EtherCat - the ethernet fieldbus.* 2017. URL: https://www.ethercat.org/en/technology.html (visited on Mar. 2017).

[66] *Ethernet Powerlink.* 2013. URL: http://www.ethernet-powerlink.org/en/powerlink/technology/how-powerlink-works/ (visited on Mar. 2017).

[67] *Ethernet Powerlink.* 2014. URL: http://openpowerlink.sourceforge.net/doc/2.0/2.0.0/d6/d51/page_powerlink.html (visited on Mar. 2017).

[68] StoneL. *Profibus Protocol - Profibus DP Profibus PA.* 2016. URL: http://www.stonel.com/en/protocols/profibus (visited on Mar. 2017).

[69] *PROFIBUS.* 2016. URL: http://www.profibus.com/technology/profibus/overview/ (visited on Mar. 2017).

[70] Linear Technology. "RS485 Quick Guide". In: *Linear Technology* (2017). URL: http://cds.linear.com/docs/en/product-selector-card/2PB_RS485fe.pdf.

[71] *Profinet Real-Time Communication.* 2017. URL: http://www.profibus.org.pl/index.php?option=com_docman&task=doc_view&gid=28 (visited on Mar. 2017).

[72] *PROFINET IO.* 2016. URL: http://www.rtaautomation.com/technologies/profinet-io/ (visited on Mar. 2017).

[73] Max Felser, Paolo Ferrariand, and Alessandra Flammini. *INDUSTRIAL COMMUNICATIONS SYSTEMS.* CRC Press, 2011.

[74] Harald Kleines et al. "Performance Aspects of PROFINET IO". In: (2008).

[75] University of Applied Sciense Duesseldorf. "CoNeT Mobile Lab 3 PROFINET on Phoenix Contact Platform". In: (2011).

[76] Zhihong Lin and Stephanie Pearson. *An inside look at industrial Ethernet communication protocols.* Tech. rep. Texas Instruments, 2013.

[77] *VLAN 0 Priority Tagging Support.* 2012. URL: http://www.cisco.com/c/en/us/td/docs/ios-xml/ios/atm/configuration/15-mt/atm-15-mt-book/atm-vlan-prty-tag.html (visited on Mar. 2017).

[78] *Configuring PROFINET.* 2016. URL: http://www.cisco.com/c/en/us/td/docs/switches/lan/cisco_ie3000/software/release/15-0_2_se/configuration/guide/scg_ie3000/swprofinet.pdf (visited on Mar. 2017).

[79] *Wireshark Go Deep.* 2017. URL: https://www.wireshark.org/ (visited on Mar. 2017).

[80] *Wireshark Profinet.* 2017. URL: https://wiki.wireshark.org/PROFINET (visited on Mar. 2017).

[81] Bradley Mitchell. *What Is a PAN (Personal Area Network)?* 2016. URL: https://www.lifewire.com/definition-of-pan-817889 (visited on Mar. 2017).

[82] Search Mobile Computing. *personal area network (PAN).* 2007. URL: http://searchmobilecomputing.techtarget.com/definition/personal-area-network (visited on Mar. 2017).

[83] William Hosch. “Bluetooth”. In: *Britannica Academic* (2017).

[84] Andrew Butterfield and Gerard Ekembe Ngondi. *A Dictinary of Computer Science.* Oxford University Press, 2016.

[85] Albert Harris III et al. “Bluetooth Low Energy in Dense IoT Environments”. In: *IEEE* (2016).

[86] *PROFIBUS.* 2016. URL: http://www.profibus.com (visited on Mar. 2017).

[87] *RASPBIAN.* 2017. URL: https://www.raspberrypi.org/downloads/raspbian/ (visited on Mar. 2017).

[88] *Integrate Revolution Pi into an industrial network.* 2017. URL: https://revolution.kunbus.com/gateways/ (visited on Mar. 2017).

[89] *Solutions for industrial communication.* 2017. URL: http://www.port.de/en.html (visited on Mar. 2017).

[90] *Codesys.* 2017. URL: https://www.codesys.com/ (visited on Mar. 2017).

A

Appendix 1 Node-Red setup

By running the following command, Node-Red will be updated to the latest release and started:

```
$ update-nodejs-and-nodered
$ node-red-start
```

To add the graph and bar of the signal strength, the dashboard user interface "node-red-dashboard" was used. Which can be found at:

```
https://github.com/node-red/node-red-dashboard
```

or by writing the following command in a terminal:

```
$ cd ~/.node-red
$ npm i node-red-dashboard
```

The flow or configuration of Node-Red can be seen in Figure A.1. The flow can be divided into two different sections, one for the Bluetooth personal emergency stop alarm and the other regarding Profinet devices. Both sections are subscribing to their respective MQTT topics, IoT/signal_strength, IoT/Alarm, IoT/signal_connected, IoT/profinet_value and IoT/red_button. The signal strength is sent to both a chart and gauge, to give a graphical representation, whereas the rest are sent to non-editable text boxes.

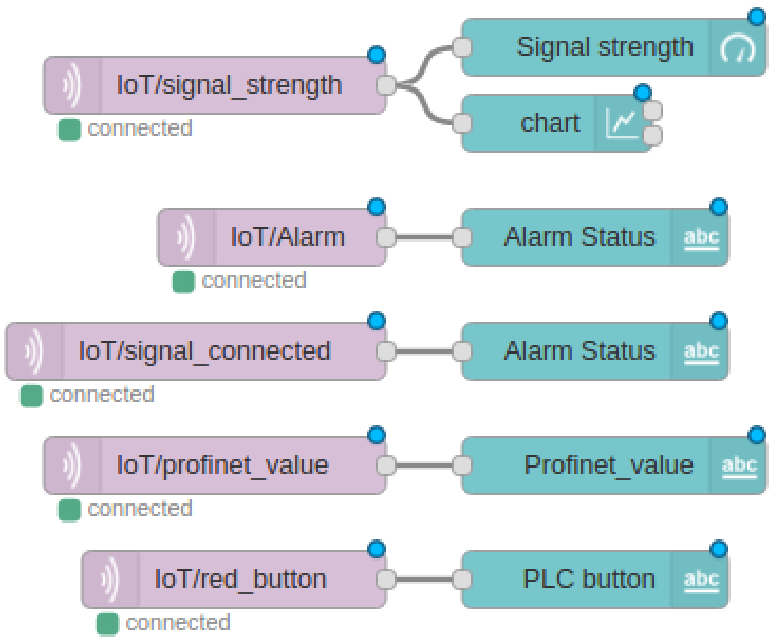


Figure A.1: Configuration layout of Node-Red

B
Appendix 2 Redis

Instructions for redis installation and setup:

```
$ sudo apt-get install redis-server
```

You will then be prompted to give the superuser password to install, another way to to get “Redis Server” installed on your machine is by visiting https://redis.io/download and downloading the latest stable release. The following command can be used to start running the server:

```
$ redis-server
```

To install Redis to be able to run directly through a terminal one can either download the script file from: https://github.com/crypt1d/redi.sh or use the following command in a new terminal window:

```
$ sudo apt-get install redis-tools
```

Once the server is running and “redis-tools” have been installed data can be read and written to the server through the command line. See example below

```
$ redis-cli SET testvariable 1
  OK
$ redis-cli GET testvariable
  "1"
```

To get access to the server from the Profinet Slave program, the recommended C client/library “hiredis” was used, which can be accessed at:

```
https://github.com/redis/hiredis
```

Other C libraries can be found at:

```
https://redis.io/clients#c
```

C

Appendix 3 MQTT

The message broker “Mosquitto” was used in the Gateway to implement the MQTT protocol. Mosquitto can be found at:

```
https://mosquitto.org/
```

Or installed and started through the command terminal with the following commands:

```
$ sudo apt-get install mosquitto
$ mosquitto
```

From the Bluetooth scripts it is possible to send MQTT messages, to activate and deactivate the alarm with the following commands:

```
$ mosquitto_pub -t 'IoT/Alarm' -m 'Alarm Deactivated'
$ mosquitto_pub -t 'IoT/Alarm' -m 'Alarm On'
```

The next part is to get the Bluetooth signal strength updated twice every second, and the following is a shorted version of how it is done in the Bluetooth script:

```
s_str=$(echo $cmdout | sed -e 's/RSSI return value: //g')
mosquitto_pub -t 'IoT/signal_strength' -m $s_str
```

D

Appendix 4 Raspberry Pi bridge setup

```
ifconfig eth0 0.0.0.0
ifconfig eth1 0.0.0.0
brctl addbr bridge0
brctl addif bridge0 eth0
brctl addif bridge0 eth1
ifconfig bridge0 up
```

E

Appendix 5 Codesys network port fix

To prevent Codesys from using all available addresses on the local machine for port 34964 a LD_PRELOAD library is used to intercept the bind() system call. Which makes it possible to force it to be bound to a specific address. A "C" program for this can be found at:

```
http://www.ryde.net/code/bind.c.txt
```

The program can be compiled with:

```
gcc -nostartfiles -fpic -shared bind.c -o bind.so -ldl
-D_GNU_SOURCE
```

To then run Codesys application with the bound address run the following command, note that you change 192.168.0.5 to whatever address is desired:

```
sudo BIND_ADDR="192.168.0.5" LD_PRELOAD=/usr/lib/bind.so
/etc/init.d/codesyscontrol
```

Credit for this solution goes to "Stack Exchange Unix & Linux" user "larsks" https://unix.stackexchange.com/a/356353/224690