Master's thesis

NTNU
Norwegian University of Science and Technology
Faculty of Information Technology and Electrical Engineering
Department of Engineering Cybernetics

Marius Lervik

System for acquisition and analyzing of data from smart meters

Master's thesis in Cybernetics and Robotics
Supervisor: Geir Mathisen
June 2019

NTNU
Kunnskap for en bedre verden

Marius Lervik

System for acquisition and analyzing of data from smart meters

Master's thesis in Cybernetics and Robotics
Supervisor: Geir Mathisen
June 2019

Norwegian University of Science and Technology
Faculty of Information Technology and Electrical Engineering
Department of Engineering Cybernetics

NTNU
Norwegian University of
Science and Technology

NTNU	**Faculty of Information Technology**
Norwegian University of	**and Electrical Engineering**
Science and Technology	**Department of Engineering Cybernetics**

Master Thesis

Candidate:	**Marius Lervik**
Course:	**TTK4900 Engineering Cybernetics, Master's Thesis**
Thesis title (Norwegian):	**System for innsamling og tolking av måledata fra smarte strømmålere**
Thesis title (English):	**System for acquisition and analyzing of data from smart meters**

Thesis description:

To modernize the power grid, most Norwegian electricity consumers have gotten smart meters installed by the start of 2019. The meters transmit data automatically to the distribution network operator using RF signals. Additionally, the smart meters provide real-time data to the consumer through a HAN port on the meter.

We want to design a system for acquisition and analyzing of data from smart meters. The data analyzing should focus on extracting information about the outdoor distribution grid.

The tasks will be:

1. Conduct a literature study concerning methods for acquiring data from in general embedded systems, and especially from smart meters. What kind of (abnormal) states of the distribution grid can data from smart meters identify?
2. Design and produce a custom embedded system needed for data acquisition.
3. Acquire data from several smart meters for at least a couple of weeks.
4. Perform an analysis and state estimation of the distribution network using the collected data.

Start date:	January 7th, 2019
Due date:	June 3th, 2019

Thesis performed at:	Department of Engineering Cybernetics
Supervisor:	Professor Geir Mathisen

Abstract

The ongoing modernization of the Norwegian electricity grid has led to the installation of smart electricity meters (AMS) for all power consumers. The AMS meters automatically send data concerning power usage directly to the distribution network operator, while personal data for the consumer has been made available through a port on the front of the meters.

In this thesis, two embedded systems for the acquisition of consumer data from AMS meters have been designed and implemented. Both systems use Wi-Fi to transfer measurement data to a cloud service for further processing and storage. Multiple units of one of the system designs were produced and successfully used for data collection from multiple sources.

Although not enough data were collected to perform a thorough analysis and state estimation of the distribution network, a significant trend in the variation in voltage values was found. The trend showed that voltage values were consistently highest during the night and lowest during the morning and evening, which corresponds to the times with the lowest and highest network load.

Sammendrag

Den pågående moderniseringen av strømnettet i Norge har ført til installasjon av smarte strømmålere (AMS) for alle strømkunder. AMS-målerne sender automatisk data til nettselskapet om strømforbruket til kunden. Personlig data er blitt gjort tilgjengelig for kunden gjennom en port på framsiden av meterne.

To systemer for å samle inn personlig data fra AMS-målere har blitt designet og implementert i denne oppgaven. Begge systemene bruker Wi-Fi til å sende data til en skytjeneste for videre prosessering og lagring. Flere enheter ble produsert av et av systemene. Disse enhetene ble brukt til å samle inn data fra flere kilder.

Datamengden som ble samlet inn ga ikke et godt nok grunnlag til å utføre en nøye analyse og tilstandsestimering av distribusjonsnettet, men en signifikant trend i variasjonen av spenningsverdier ble oppdaget. Trenden viste at spenningsnivået var konsekvent høyest på natta og lavest på morgenen og kvelden. Disse tidspunktene tilsvarer tidene det er minst og størst belastning på distribusjonsnettet.

Preface

This thesis is submitted in fulfillment of the degree of Master and Science at the Department of Engineering Cybernetics at the Norwegian University of Science and Technology. The work performed in this master thesis continues to build on the findings and work of the specialization project in TTK4550.

One of the systems developed is based on the work done by me in the specialization project. Excluding the work from the specialization project, all the work presented in this thesis is performed by me between the start and due date of the thesis. The implementation chapter specifies any use of external libraries or SDKs in the implementation of the software.

Working with this thesis has been fun and challenging, and it has given me valuable experience in developing embedded software and hardware, which will surely be useful later on.

Acknowledgments

I want to thank the supervisor Geir Mathisen for helping me with the thesis, and providing an AMS meter for testing and development, which proved to be very useful. I also want to thank Mats Kornelius Karlsen and Ingvild Skaftun for using the developed embedded system to collect data, which hopefully has been somewhat useful for their thesis work. Lastly, I want to thank the employees at the electronic workshop at the Department of Engineering Cybernetics for letting me use their equipment for working with the embedded hardware.

Approved: G Mathisen
Prof. Geir Mathisen
Supervisor of thesis

Approved: Marius Lervik
Marius Lervik
Author of thesis

Table of Contents

List of Tables

List of Figures

Abbreviations

AMS	=	Advanced Metering System
HAN	=	Home Automation Network
MCU	=	Microcontroller Unit
PCB	=	Printed Circuit Board
IC	=	Integrated Circuit
TLS	=	Transport Layer Security
IoT	=	Internet of Things
AP	=	Access Point
UART	=	Universal Asynchronous Receiver-Transmitter
API	=	Application Programming Interface
RF	=	Radio Frequency
ESD	=	Electrostatic Discharge
EEPROM	=	Electrically Erasable Programmable Read-only Memory
SDK	=	Software Development Kit
LDO	=	Low-dropout
ISR	=	Interrupt Service Routine
VPS	=	Virtual Private Server

Chapter 1

Introduction

1.1 Background

1.1.1 Electric power systems

An electric power system can be roughly divided into three parts, power generators, the transmission system, and the distribution system. An illustration of a traditional electric power system is shown in fig. 1.1. The complete electric power system connects many power generators to the transmission system, which connects to distribution systems where most consumers are located.

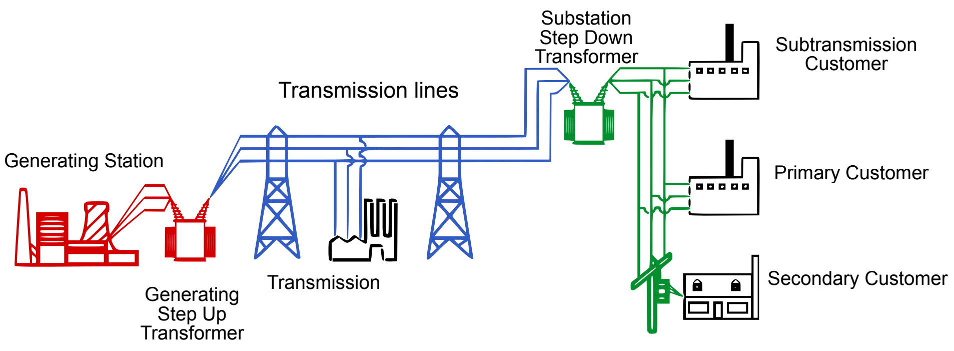


Figure 1.1: Traditional electric power system [20].

The generators produce electric power from mechanical power, for instance, hydropower or thermal power. In Norway, the majority of power generation comes from hydropower. In order to transfer the electrical power over vast distances, the voltage is transformed into high voltage, typically in the range of 100kV to 500kV, although transmission lines with a voltage higher than 1000kV exist. The high voltage results in smaller losses and is essential when transferring power over long distances. Although high-voltage DC transmission systems exist, most transmission lines use high-voltage AC.

For the Norwegian power network, the transmission system consists of two parts, a central transmission network (132kV-420kV) and a regional transmission network (66kV-132kV) [16]. The central transmission network is mostly owned by *Statnett* and it connects large power producers and the regional networks from all over the country. The regional transmission networks are owned by local network companies and serve as connections between the central transmission network and the distribution networks. Some smaller producers and industries that require a large amount of power are connected directly to regional networks [16].

The distribution network is the final stage in the delivery of electrical power, and it is where power is delivered to most consumers. The distribution network consists of both high voltage (11kV or 22kV in Norway) and low voltage (230V or 400V in Norway) lines. The typical low voltage parts of the distribution network are radial connections, which means there is a single power source delivering power to multiple consumers. A radial system is illustrated in fig. 1.2 where the two circles symbols a transformer and the rectangles are consumers. Although other topologies exist, the radial distribution is the most common topology used to connect consumers to the distribution network. A radial distribution system can consist of one or more main radials with a larger power transfer, which branches into smaller radials.

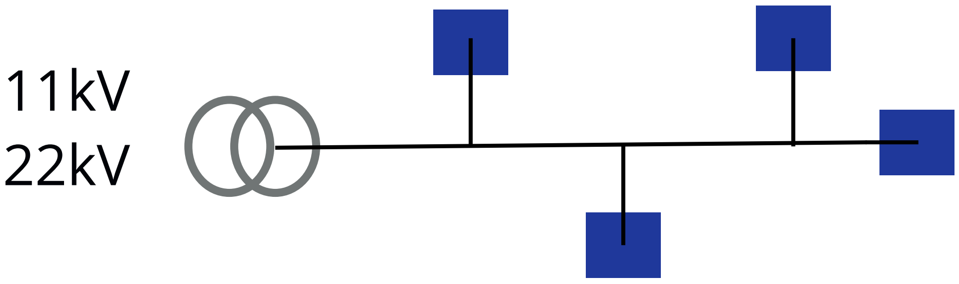


Figure 1.2: Radial power distribution.

The radial topology will typically result in a small voltage drop along the line, where the consumers furthest away from the transformer will have the lowest voltage values. This relationship becomes more complicated if the radial has multiple branches or if some consumers are producing power and delivering this power into the network. Customers producing power and feeding this into the network is becoming more common as a result of research, commitment, and awareness concerning renewable energy.

The transformer station decreasing the voltage into low voltage (230-400V) has traditionally been the last point where the distribution network operator can monitor measurement values like voltage, current, power, and energy in real time.

1.1.2 Smart grids

Traditionally there has been a one-way power transfer from the generation point to the consumers. In addition to only being unidirectional, a considerable amount of power is lost along the transmission lines, and some of the generation capabilities exist only to be able to meet peak demand [8]. The development and progress made in the fields of information technology and big data analysis allow for a different network structure commonly referred to as a smart grid. The smart grid is a grid which allows for the multi-directional transfer of energy and data as opposed to the traditional electricity grid. A smart grid can have a wide variety of power generation options and can result in less waste of power by distributing power in a more efficient way than the traditional electric power system.

As renewable energy sources are getting more common, the amount of customers delivering energy to the network is increasing. Most of these customers are located in the distribution network, which is also where most of the power outages and disturbances occur [8]. A smart grid contains several power generation options, including distributed and intermittent power generation. Hence, a smart grid would help to mitigate problems with power outages and disturbances by using these additional power generation sources.

Closely related to the concept of smart grids are *smart microgrids*, commonly called microgrids. A microgrid can be defined as a local network of distributed energy systems, including both generators and consumers, where the local network can function both when connected or disconnected to the broader electricity grid [8]. It follows that a microgrid must have some source of distributed power generation, which is often a renewable energy source. Additionally, a microgrid must handle supplying emergency power and transitioning between being connected and disconnected to the electricity grid. Multiple microgrids can be interconnected with each other to form a smart grid where data and electric power are transferred between the microgrids.

The main motivational factors for developing smart grids are the need to handle increased complexity in the grid and to be able to have a complete overview and control of the grid [19]. This includes the ability to monitor the network in real-time, where the analyzed information can be used to create control systems in order to optimize the operation of the grid. A complete overview and control of the grid require a large number of sensors and control components to be placed throughout the distribution network. The sensors must be able to measure a multitude of parameters, for instance, power consumption, voltages, and currents.

In addition to sensors at essential points in the distribution network, smart metering equipment at the consumer is beneficial for both the network operator and the consumer. Electricity meters with two-way communication to the network operator are called smart meters. Consumption data from the meter can be presented to the consumer in real-time, which makes it possible to monitor and regulate electricity usage. Moreover, smart meters automatically send consumption data and other event messages to the network operator, which leads to more accurate billing of customers and improved monitoring capabilities.

1.1.3 Advanced Metering System

In Norway, smart meters named Advanced Metering System (AMS) have been installed for almost all electricity customers by the start of 2019. The meters send measurement and power usage data to the network operator at least once an hour [17]. In addition to measurement data, the meter logs some events. These events include loss of power, voltage unbalance, over-voltage, and under-voltage [3]. The meters are also able to detect grounding faults, and immediately send a corresponding alarm to the operator [3]. There are three suppliers of the Norwegian AMS meters, Kaifa (Nuri/Kaifa), Kamstrup and Aidon.

The AMS meters are equipped with a HAN-port with an RJ45 (8P8C) connector. The consumer can get access to consumer data and measurement values through this port. A list of the measurement values available for Kaifa meters is shown in Appendix A where the contents of the different meter messages are divided into different lists. List 1 only contains an active import power measurement and is the most frequent message sent. List 2 is sent every tenth second and contains more values including, active and reactive power (both import and export), the instantaneous current draw for the different phases and voltage values. The final message containing List 3 is sent every hour and contains all the values in List 2 in addition to hourly cumulative energy usage values. The HAN specification is similar for Kaifa and Aidon, while Kamstrup have omitted List 1 in their specification. Consequently, Kamstrup meters only send out information every tenth second as opposed to Kaifa (every 2s) and Aidon (every 2.5s) [2].

The measurement values sent out depends on the meter type. For single-phase meters, which are more common in apartment blocks, there is only one current and one voltage value available, as opposed to three-phase meters, which have three current and voltage measurement values (one for each phase). In order to utilize the HAN data, one has to contact the network operator and request the activation of the HAN port as it is not activated by default.

1.2 Limitations

The data collection performed in this thesis was not extensive enough to perform a comprehensive analysis of the distribution network. Ideally, data should be collected from consumers connected to the same transformer/radial in order to perform a more thorough analysis of the state of the surrounding distribution network. Only three systems collecting AMS data were initially set up. Four additional systems were set up very late during the thesis work, which resulted in the data from these meters not being analyzed because of time limitations.

1.3 Thesis structure

Chapter 2 Literature Review contains a literature review on error detection and distribution network topology estimation using smart meter data, existing solutions for logging AMS data, and protocols for use with IoT.

Chapter 3 Relevant Standards and Protocols describes standards and protocols relevant to the thesis work and the AMS meters. The purpose of this chapter is to present an overview of all standards related to AMS meters, and it is not necessary to read in its entirety as it contains very detailed information.

Chapter 4 Specification and Design presents a functional specification for the hardware, software, and the complete system. The chapter contains a high-level design of the hardware and data collection scheme. Acceptance criteria for the complete system are included in this chapter.

Chapter 5 Implementation contains implementation details for both hardware and software for the two embedded systems implemented. Implementation details for the cloud service and data collection are also included.

Chapter 6 Testing and Results describes the testing procedure, testing results, and the setup for the data collection. The chapter presents graphs of the collected AMS data.

Chapter 7 Discussion discusses the results and performance of the system compared to the acceptance criteria. It contains a discussion of the graphs presented in the results section. Additionally, some inconsistencies with the AMS standards and potential improvements are discussed.

Chapter 8 Conclusion concludes the thesis work, results, and findings.

Chapter 9 Further work contains a bullet point list of suggested further work.

Appendix A contains the Kaifa HAN specification. **Appendix B** and **Appendix C** contains complete schematics and component lists for the implemented embedded systems. Lastly, **Appendix D** contains an email received from the Norwegian Electrotechnical Committee (NEK).

Chapter 2

Literature Review

2.1 Error detection using smart meters

This section contains a literature review on the usage of smart meter data to detect errors and faults in the low voltage part of the distribution network. A literature review has been performed on two different topics, high impedance faults and network topology errors at the distribution network operator.

2.1.1 Validation of distribution system topology

A distribution network operator must have information about the topology of the low voltage distribution network. This information is stored in a Network Information System (NIS), which consists of two other systems, namely the Geographic Information System (GIS) containing the geographical location of all objects, and the Customer Information System (CIS) containing information about the customers and other relevant information [3]. The topology information consists of the electrical connections of network stations, conductors, and customers. Additionally, information concerning cable properties and information about the position of electrical switches are also part of the topology information.

It is essential that the topology information stored in the NIS is correct in order to utilize the potential of smart meter data and other measurement data collected, as incorrect topology information could result in incorrect results from computations and algorithms using the smart meter data. There can be multiple errors in the topology information stored in the NIS. One common type of error is that the NIS contains wrong connection information, for instance, that a customer is connected to an incorrect radial. This error is illustrated in fig. 2.1, which shows that in the NIS, the red customer is connected to the transformer NS1, while it is in reality connected to NS2.

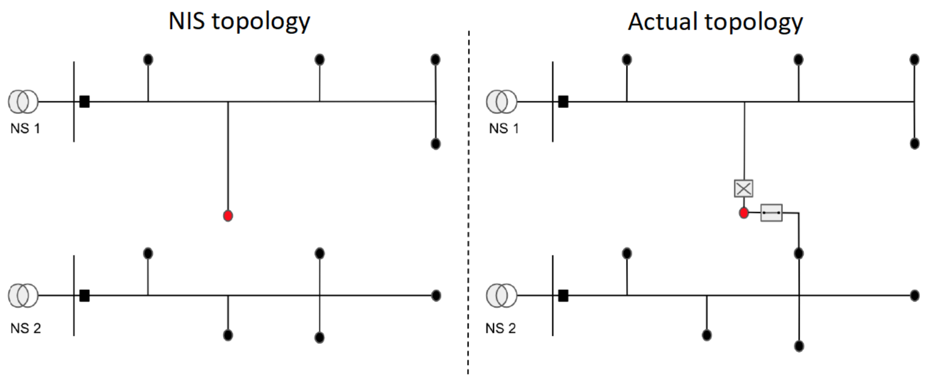


Figure 2.1: NIS topology error showing an incorrect connection [3].

Other NIS errors include wrong electrical switch position, wrong information about cable lengths and customers not connected to the network. For customers not connected to the network, the error is only in the NIS, while in reality, they are connected. There can be multiple sources for these errors, for example, data import errors, lacking information, or simply an error with manual entry of information.

Multiple methods have been developed to estimate and validate the topology information for low voltage distribution networks. A method called Distribution system Topology Estimation (DSTE) is presented in [10]. The algorithm uses voltage values and current values, both real and imaginary, to estimate impedance and connections. It is based on linearized voltage drop approximation and linear regression in order to estimate the topology of the network downstream of a transformer. Real and imaginary current values are not available from the AMS meters. However, they may be estimated using the other measurement values available.

A less complex method than the DSTE method is the voltage drop method explained in [3]. The method is based on the fact that there will be a voltage drop outward in a radial where the highest voltage values are close to the transformer. An example is shown in fig. 2.2, which shows five customers connected to a radial. Based on the cable lengths and locations in the figure, one would expect that $V_1 > V_2 > V_3 > V_4 > V_5$. If this is not the case, there is a possibility that there is a topology error in the NIS. There are some limitations to this method. If there is a consumer producing power and delivering this to the network, the voltage values may not be as expected from using the voltage drop method.

Other methods for validating network topology are presented in [3], where more advanced algorithms using these methods are also described. The algorithms developed to validate topology are split into three categories where each category uses some different basic properties, which are, sum, affiliation, and sequence.

For the sum property, an energy balance calculation is used to check that the amount of customers in the low voltage circuit is correctly documented in the NIS. The method

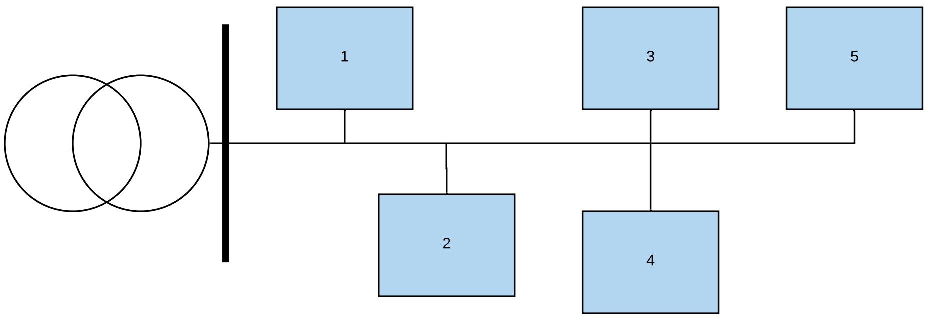


Figure 2.2: Low voltage circuit with five consumers connected.

described summarizes the energy usage of all customers and checking this against the measured energy consumption at the transformer station. For the affiliation property, two methods are used, the ground fault method and the Total Harmonic Distortion (THD) method to check that a customer belongs to the circuit documented in the NIS. Lastly, for the sequence property, the voltage drop method was used to validate that the sequence of the customers is correct.

The thesis in [3] uses the data sent to the network operator, however much of this data is also available for the customer through the HAN port. Additionally, the HAN data contains exact values sent out every tenth second, while the data to the network operator contains an average in addition to maximum and minimum values [3]. If the data from the HAN port should be used to detect errors, the data collection must be done with a separate system from the data sent to the network operator. If this data were to be used to detect errors in real-time, this separate data acquisition system has to be reliable.

2.1.2 High impedance fault detection

A high impedance fault (HIF) is the result of when an energized conductor comes in direct contact with a high impedance surface, such as concrete, asphalt, or tree branches [23]. Conventional protection mechanisms have a hard time detecting these errors since the high impedance will limit the amount of current, and as a result, over-current protection mechanisms, such as fuses and over-current relays will not easily detect the fault [24].

Although HIFs represents little threat to the electrical equipment, they represent a threat to the surroundings. It is typical for a HIF to be accompanied by an electric arc. This effect is shown in fig. 2.3. The arc ignition occurs if the voltage of the conductor in contact with the surface becomes sufficiently high and exceeds a voltage threshold required to create an electric arc. Since most distribution systems use AC, the magnitude of the voltage varies, and multiple arc re-ignitions and extinctions can occur with a high impedance fault [9]. If a HIF occurs in a hazardous area, the accompanying electric arc could case result in fire or explosion. Therefore, it is important to detect and handle high impedance faults.

Figure 2.3: High Impedance Fault causing an electric arc [10].

Several existing HIF detection techniques are summarized in [9] where various measurement values are used. The simplest of the techniques uses voltage and current measurements to detect faults. The current measurement is most effective in distribution networks where the frequency components of the current waveform are not significantly attenuated [9]. In order to develop more robust detection techniques, a higher order analysis have to be performed on the measurement values. The electric arc associated with a HIF results in high and low-frequency components in the current spectrum, which can be used for detection of HIFs. Time-Scale analyses (wavelet transform) is a common technique used in many approaches to detecting HIFs [9]. This method uses both frequency information and the time information where those frequencies occurred, to analyze the current waveforms. Some other methods also exist, including using classifiers and artificial neural networks [9].

In [23], a method for detecting HIFs using smart meters is presented. The method suggests detecting a voltage unbalance which will occur downstream from a HIF, using a three-phase smart meter. The method proposed uses a method called the symmetrical components method. With three-phase power, both voltage and current consists of three components which can be represented as three phasors. In a balanced three-phase power system, the phasors have equal magnitudes and are 120 degrees apart. In an unbalanced power system, the magnitudes of the phasors are different, and they are not exactly 120 degrees apart. Decomposing the phasors into symmetrical components is used for this method as shown in eq. (2.1), where the 0, 1, and 2 components are called zero, positive and negative sequence components respectively. The sequence components are symmetrical and have a 120 degrees difference in phase angle.

$$\begin{bmatrix} V_a \\ V_b \\ V_c \end{bmatrix} = \begin{bmatrix} V_{a,0} \\ V_{b,0} \\ V_{c,0} \end{bmatrix} + \begin{bmatrix} V_{a,1} \\ V_{b,1} \\ V_{c,1} \end{bmatrix} + \begin{bmatrix} V_{a,2} \\ V_{b,2} \\ V_{c,2} \end{bmatrix} \tag{2.1}$$

The method proposed in [23] uses symmetric voltage components to calculate a degree of unbalance. This is done by calculating the relationship between the negative sequence voltage (V_2) and the positive sequence voltage (V_1). The equation for calculating voltage unbalance in percentage is shown in eq. (2.2).

$$K_{\%} = \frac{|V_2|}{|V_1|} \tag{2.2}$$

For smart meters, the phase voltage, line voltage, or possibly both are available as a measurement value. The phase voltage is the voltage between a phase and a neutral while the line voltage is the voltage between two phases. If the line voltages are available, the voltage unbalance can be calculated with the equations in eq. (2.3) and eq. (2.4) found in [23]. The subscripts on the voltages indicate which line voltage to use where a, b, and c are the three phases.

$$K_{\%} = \sqrt{\frac{1 - \sqrt{3 - 6\beta}}{1 + \sqrt{3 - 6\beta}}} \tag{2.3}$$

$$\beta = \frac{V_{ab}^4 + V_{bc}^4 + V_{ca}^4}{(V_{ab}^2 + V_{bc}^2 + V_{ca}^2)^2} \tag{2.4}$$

Since the smart meters are installed at consumers, a selection of smart meters could be used for the detection and identification of HIFs. Knowledge of meters and their respective transformers are needed to detect a HIF. The calculated voltage unbalances and knowledge of the meters respective transformers, both upstream and downstream of where the fault occurred, can be used to develop a methodology to detect and locate the fault. This is possible since a HIF will cause a high voltage unbalance downstream from the location of the fault.

A case study using simulated data was performed in [23]. A case is shown in fig. 2.4, where SE is a distribution substation, TR are transformers and a smart meter is placed downstream of every transformer. In the case of fig. 2.4, a HIF has occurred between point 4 and 5. The method presented in eq. (2.3) and eq. (2.4) resulted in a unbalance factor of approximately 50-99$\%$ for the smart meters located downstream of TR3 and TR4 using simulated data. The reason for the large range in the unbalance factor is that different parameters were used in the simulation. For the other smart meters, the unbalance factor was significantly lower at approximately 1-5$\%$.

Another method for detecting HIFs using smart meter data is suggested in [6]. This method detects HIFs by using an Even Harmonic Distortion Index calculated over the voltage waveforms. This method requires the smart meters to be able to perform an even harmonic

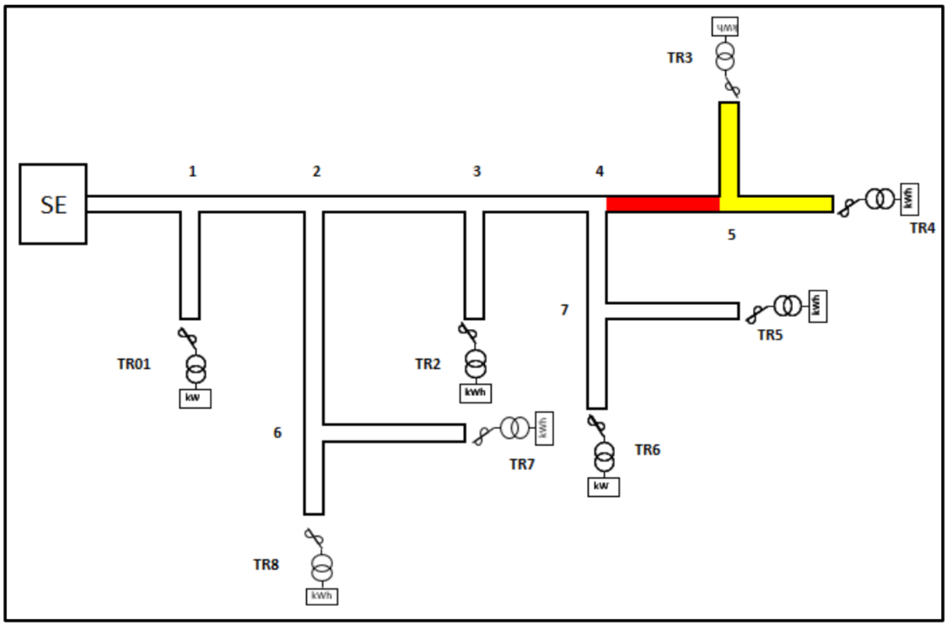


Figure 2.4: Location and detection of a HIF using smart meter data [23].

estimation algorithm, as opposed to the calculation in eq. (2.3), which only requires values already available from smart meters.

2.2 Existing AMS logging solutions

At the time of writing [14] there was no commercially available solution for logging AMS data through the HAN port. Since then, a device available for purchase has surfaced, called Tibber Pulse from Tibber. Tibber Pulse is made to work with smart meters from all three suppliers in Norway. The only connection needed is a standard twisted pair ethernet cable between the device and the smart meter.

The device uses Wi-Fi to upload data to a cloud service controlled by Tibber. The user can get this data by using a mobile app from Tibber or by using an API. Only power and energy values are made available for the consumer using this device.

From images posted on a forum [21], it seems like the Tibber Pulse does not make use of an M-Bus slave transceiver chip to receive data from the M-Bus lines. Furthermore, it looks like a large capacitor, rechargeable battery or a supercapacitor is used to store electric energy for backup power. It can also be seen from the images that the Tibber Pulse uses an ESP32-WROOM-32D module from *espressif* as a processing unit [21]. This module contains an ESP32 chip as the MCU and an antenna which can be used for Wi-Fi, Bluetooth, and Bluetooth Low Energy.

The module draws power solely from the HAN port of the smart meter, which means that no external power supply is necessary. For the Kamstrup smart meters, the maximum power draw is only 144mW with a maximum current draw of 6mA [2]. The ESP32 chip uses between 180-240mA when transmitting using Wi-Fi [7]. If all RF capabilities of the unit are turned off and the CPU is on, the current draw is approximately 20-31mA according to [7]. This very high current consumption compared to the maximum current draw of the smart meters must mean that the Tibber Pulse device utilizes some of the sleep modes available on the MCU. Furthermore, the sending frequency of the device must be low enough such that enough time has passed in order to charge the local power storage on the board.

2.3 A survey on protocols for use with IoT devices

The IoT has been enabled by the latest development and research into smart sensors, communication technologies, and communication protocols. An overview of technologies, protocols, and possible applications related to IoT is given in [1]. The paper provides a thorough summary of several different application layer protocols, and it explores the relation between IoT and other emerging technologies such as big data analytics, cloud computing, and fog computing.

Several application layer protocols including Constrained Application Protocol (CoAP), Message Queue Telemetry Transport (MQTT), Extensible Messaging and Presence Protocol (XMPP), Advanced Message Queuing Protocol (AMQP), Hyper Text Transfer Protocol (HTTP) and Data Distribution Service(DDS) are presented and compared. Both MQTT and AMQP are based on sending messages which have to go through a server (broker). AMQP contains more advanced functionality than MQTT, for instance, reliable queuing and flexible routing. Despite its name, MQTT does not support queues. Hence, it is a more lightweight protocol than AMQP.

A comparison of the overhead and performance of HTTP and MQTT is presented in [25]. The paper concluded that the performance of MQTT is superior to HTTP, particularly when the number of devices is large, the overhead and performance of HTTP are significantly worse than MQTT. The overhead is illustrated in fig. 2.5, which shows the total bytes sent compared to the payload length sent for a different number of devices.

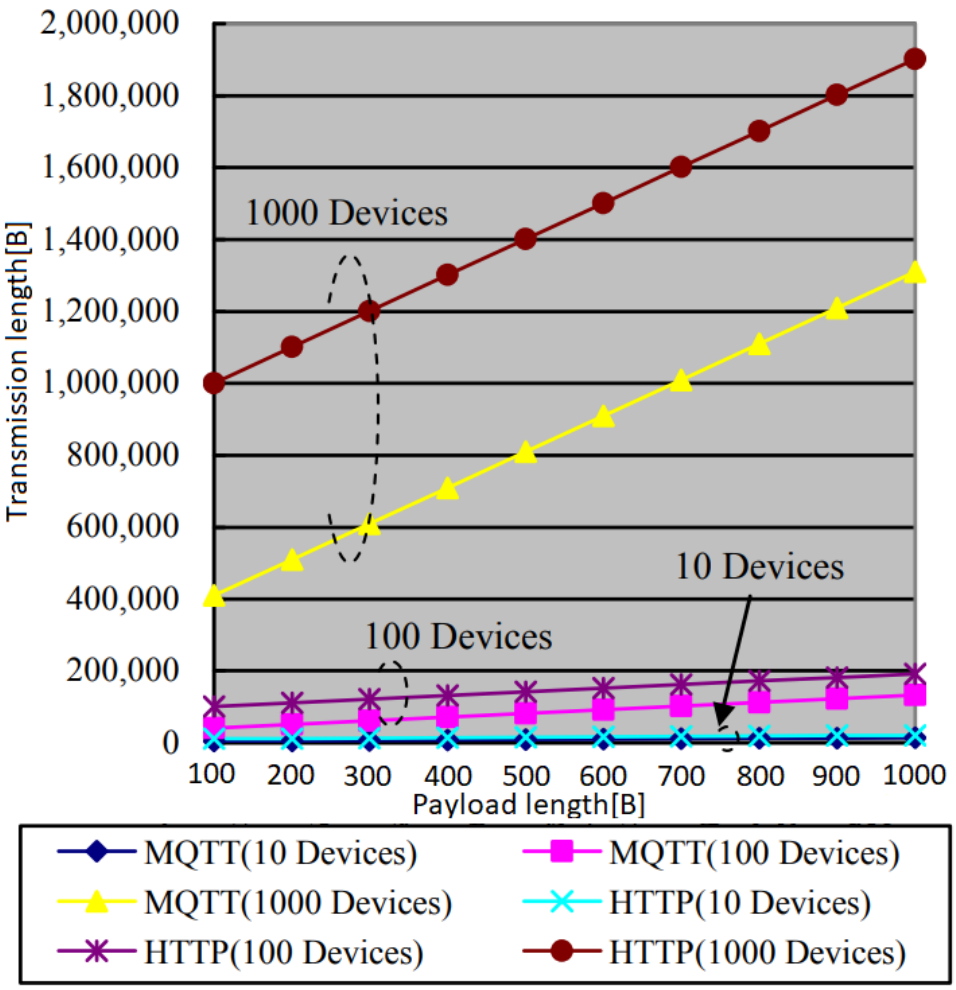


Figure 2.5: Relationship between payload size and transmitted bytes for the MQTT and HTTP protocols [25].

Chapter 3

Relevant Standards and Protocols

This chapter contains descriptions of various standards and protocols relevant to the work done in this thesis. This includes standards and protocols adhered to by the AMS meters and protocols used with the embedded system and cloud solution implemented in this project. Some of the content in this chapter is a summary of theory presented in [14], including section 3.1, section 3.2, section 3.3 and section 3.4. A detailed understanding of the standards related to AMS meters is not required in order to understand the rest of the thesis.

3.1 Standards for AMS meters

The AMS meters adhere to various standards and protocols summarized in fig. 3.1. The physical layer protocol used in the AMS meters is the physical layer of the M-Bus standard described in EN 13757-2. The data link layer protocol is the HDLC protocol as described in IEC 62056-46. Finally, the meters adhere to various standards from the DLMS/COSEM set of standards. The international standard version of the DLMS/COSEM standard is described in the IEC 62056 series of standards.

3.2 M-Bus

M-Bus (Meter-Bus) is a European standard developed for the remote reading of utility meters. It is included in the EN 13757 standard for communication systems for utility meters. The M-Bus standard consists of standards for the physical layer, data link layer, application layer, and an optional network layer. An old documentation of the standard is freely available at [22]. Furthermore, a wireless version of the standard is specified in EN 13757-4.

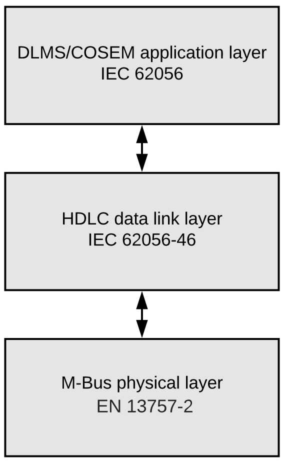


Figure 3.1: Illustration of the standards used in AMS meters [14].

M-Bus is based on a master-slave topology consisting of one master and several slaves where all slaves are connected to the same transmission lines. In the case of the AMS meters, the meter itself acts as a master while any device connected must be an M-Bus slave device. For the AMS meters, only the physical layer of the M-Bus standard is used, thus only the physical layer of the standard is presented in this section.

3.2.1 Physical layer

The physical layer of the M-Bus standard is a bus consisting of two wires, where transmission is possible in one direction at a time, that is either master to slave or slave to master. In the case of data transfer from master to slave, a logical 1 is represented by a voltage difference of 36V, and a logical 0 is represented by a 24V difference. This choice of voltage levels allows powering of slave devices directly from the bus lines. Data can be transferred from slave to master by modulating the current consumption of the slave. A logical 1 is represented by a constant current draw of up to 1.5mA whereas a logical 0 is represented by an increased current drain by the slave of additional 11-20mA. This bit representation scheme is illustrated in fig. 3.2. As seen from the figure, the increased current draw from the slave transmitting a 0 to the master causes a small voltage drop on the transmission lines.

The quiescent (inactive) bus state is when the bus voltage is 36V, and the current draw of slaves is less than 1.5mA. As a result of cable loss, the voltage levels at the slave will be lower than 24-36V. Hence, slaves must detect a voltage difference of approximately 12V and not the absolute voltage in order to differentiate between a logical 0 and 1.

The physical layer makes some demands on the data link layer that will be used. It requires half-duplex asynchronous serial transmission with a baud rate of 300-9600 Baud.

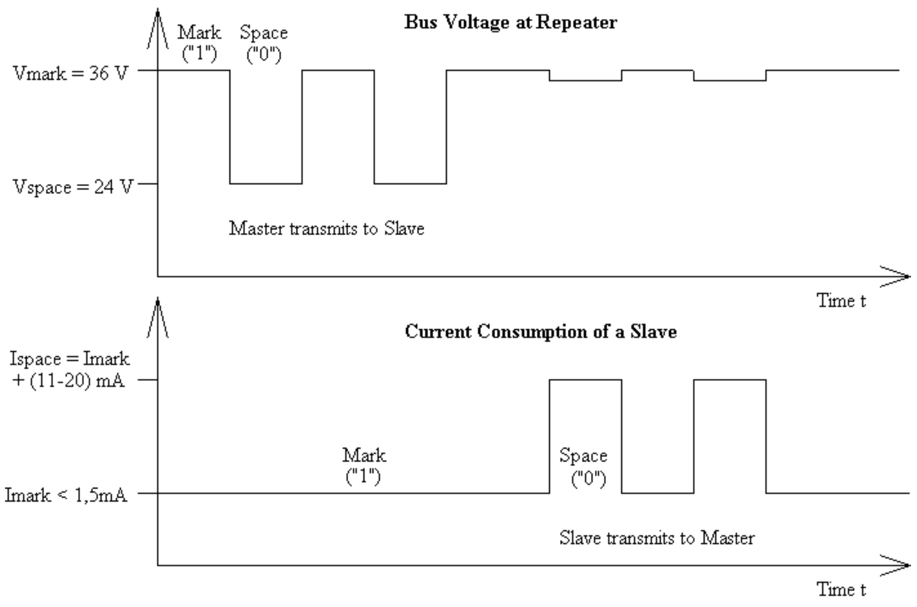


Figure 3.2: M-Bus physical layer bit representation [22].

Furthermore, there must be a master-slave structure where the slaves cannot communicate with each other, and at least every eleventh bit sent on the bus should be a logical 1.

3.3 HDLC

High-Level Data Link Control (HDLC) is a data link layer protocol developed by the International Organization for Standardization (ISO). Although multiple network topologies are possible, HDLC is mainly used for point to point connections.

The version of HDLC described in this section is the version described in IEC 62056-46, where an excerpt of this standard is available in [5]. The IEC 62056-46 standard defines that this specification supports asynchronous start/stop transmission, with 1 start bit, 8 data bits, no parity bit and 1 stop bit. These conditions satisfy the demands by the physical layer of the M-Bus protocol presented in section 3.2.1.

The HDLC protocol consists of two sublayers, the Logical Link Control (LLC) upper sublayer and the Medium Access Control (MAC) sublayer. The LLC sublayer is based on the ISO/IEC 8802-2 standard, and the MAC sublayer is based on the ISO/IEC 13239 standard. The LLC sublayer acts as an interface between the MAC sublayer and an optional network layer, making it possible for several network layer protocols to coexist in a network. The MAC layer is the layer that ensures the link layer connection and contains the data payload. Only the MAC sublayer format is presented, as the LLC sublayer is not important for AMS messages.

3.3.1 MAC layer frame format

The MAC layer for DLMS/COSEM uses the HDLC frame format type 3 defined in ISO/IEC 13239. This frame format is shown in table 3.1 where the field length is shown in bytes in the second row. The role of the fields is explained below the table.

Flag	Frame format	Dest. address	Src. address	Control	HCS	Information	FCS	Flag
1B	2B	1-4B	1-4B	1B	2B	xB	2B	1B

Table 3.1: HDLC MAC layer frame format.

Flag field

Contains a start/stop byte for frames. The value of this field is always 0x7E. If two or more frames are transmitted consecutively, a single flag is used as the start and stop byte in between the frames.

Frame format field

This field contains three sub-fields. The first sub-field is the frame format type and consists of the first four bits. In the case of DLMS/COSEM, the frame type is 3. The fifth bit is a segmentation bit. The 11 remaining bits contain the length of the frame in bytes excluding start and stop flags.

Destination and source address field

Contains the destination and source address of the frame.

Control field

The control field indicates the type of commands or responses, and it contains sequence numbers where appropriate.

Header check sequence (HCS) field

The header check sequence is a checksum that is applied only to the header. That is the bits between the start flag and the HCS. The HCS is calculated in the same way as the frame check sequence.

Information field

This field contains the data that is sent using the HDLC protocol if a data frame is sent. The LLC sublayer frame is included at the start of the information field.

Frame check sequence (FCS) field

The frame check sequence contains a 16-bit checksum calculated on the entire length of the frame excluding the start and stop flags. An example of an implementation of the 16-bit FCS is found in [12]. The FCS was originally designed to be implemented in hardware. Hence it can be calculated bytewise on a transmitted or received byte stream. The receiver has no way of determining when it has finished calculating the FCS until it detects the stop flag. Consequently, the FCS is designed so that a particular pattern results when the FCS calculation operation passes over the complemented FCS in the frame. The frame content can then be verified by checking that the final result of the FCS calculation is equal to a specific value defined in [12].

3.3.2 Frame transmission

HDLC frames can be transmitted over both synchronous and asynchronous communication links. For both synchronous and asynchronous framing, measures are added to avoid sending a start/stop flag in the middle of a message. The method of solving this is different for synchronous and asynchronous communication, and it is described in ISO/IEC 13239.

In the case of synchronous transmission, the transmitter uses bit stuffing, which works by inserting a 0 bit following all sequences of five consecutive 1 bits. The receiver must examine the content of the frame and discard any 0 bit which comes after five consecutive 1 bits. Consequently, the sequence of bits which forms the flag byte will never be present in the message.

Asynchronous framing sends one byte at a time and utilizes byte stuffing, alternatively called control-octet transparency. The standard defines a control octet (byte) as 0x7D. Every occurrence of the control octet or flag byte is replaced with the control octet followed by the original byte with bit number 5 complemented, where bit number 0 is the LSB. This method results in a 0x7E being replaced by 0x7D 0x5E and a 0x7D being replaced by 0x7D 0x5D. Consequently, if a receiver receives a control octet, it knows that this byte is not part of the message, and the fifth bit in the next byte received must be complemented in order to get the correct value.

3.4 DLMS/COSEM

The EN 13757-1 standard together with IEC 62056, is the most widely accepted international standard for utility meter data exchange, and is commonly referred to as DLMS/-

COSEM. The DLMS/COSEM set of standards is large and complex, and only a small part of it is presented in this section. The information presented here is mostly based on the excerpts in [5] and [4]. A deeper understanding of the DLMS/COSEM standards is not required in order to understand the rest of this thesis.

3.4.1 Specification

Device Language Message Specification (DLMS) is a concept for structured modelling of the interface of a utility meter. Companion Specification for Energy Metering (COSEM) sets rules for data exchange with energy meters, including specifications that define the transport and application layers of the DLMS standard. A three-step approach is specified by the standard where the steps are as follows.

1. Modelling
2. Messaging
3. Transporting

The modelling step covers the model of the metering equipment and rules for data identification. This includes specifications of COSEM interface classes, and the OBIS object identification system, which is a naming system used on COSEM interface objects.

The messaging step covers the communication services and the protocols used to map elements of the data model from the modelling step to an application layer message. Finally, the transporting step covers the services and protocols for transportation of messages.

3.4.2 Architecture

The DLMS/COSEM standard is based on a client-server architecture where information is exchanged between meters and data collection systems. The metering device has the role of a server while a data collection system is a client. A client can send service requests to the server, which will respond with a service response. Furthermore, the server can send unsolicited service requests to the clients containing information about events, or send data on some conditions, such as a preconfigured time interval. An illustration of the client-server model and communication is shown in fig. 3.3.

3.4.3 OBIS

An OBIS code is used to identify data items in smart meter equipment, which provides a unique identifier for all data within a meter. The data includes measurement values and also includes abstract values used for obtaining information or configuration of the metering equipment. OBIS codes consist of the six value groups A-F defined in table 3.2, where value groups D-F are mostly used for further classification of abstract data.

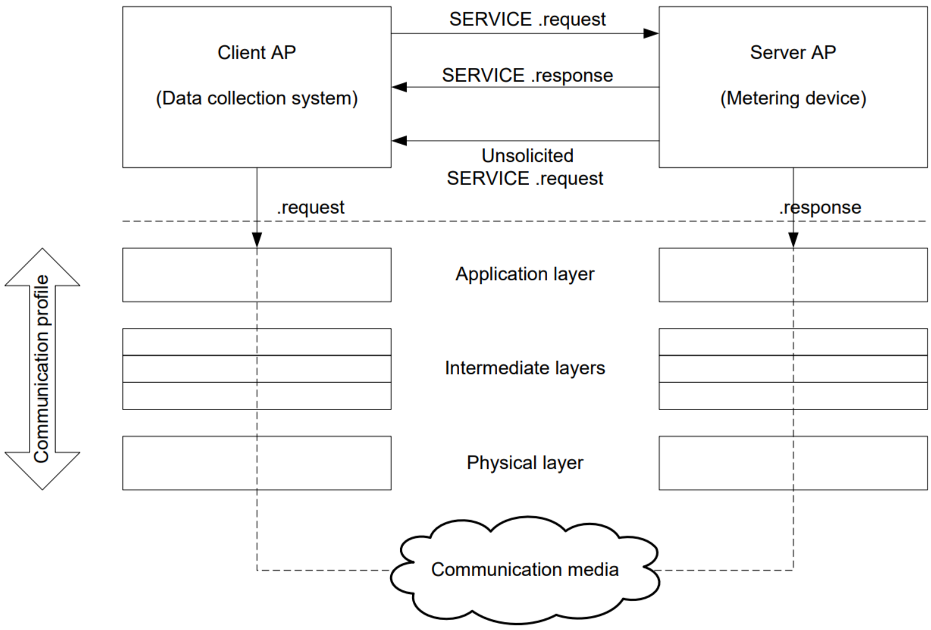


Figure 3.3: DLMS/COSEM client-server model [5].

Group	Use of value Group
A	Identifies the media (energy type) to which the meter is related. Data not related to a media is handled as abstract and represented by a 0 in this group.
B	Generally used for identifying the measurement channel number where a meter has several inputs for the same energy type. Used to identify data from different sources. It may also identify the communication channel or some other elements. The definitions in this group are independent of value group A.
C	Identifies abstract or physical data items related to the information source. Examples are current, voltage, power, and temperature. The definitions depend on the value in group A.
D	Identifies types or the result of the processing of physical quantities identified by values in value groups A and C, according to various specific algorithms.
E	Identifies further processing or classification of quantities identified by value groups A-D.
F	Identifies historical values of data identified in value groups A-E according to different billing periods. If this is not relevant, this value group can be used for further classification. Set to 255 if not used.

Table 3.2: OBIS code structure and use of value groups [4].

An example of OBIS codes is shown in Appendix A, which shows the OBIS codes for the Kaifa HAN specification together with the corresponding meter measurement values.

3.5 MQTT

MQTT stands for Message Queuing Telemetry Transport and is a publish-subscribe-based message protocol. It is a lightweight protocol designed with simplicity in mind. As a result, it is well suited for settings where a small code footprint is required, and network bandwidth is limited. It follows that MQTT is a protocol well suited for IoT and embedded devices. There are two major versions of the standard, version 3.1.1 and version 5.0. The information presented here is based on MQTT version 3.1.1. Documentation for both versions is available at [18].

3.5.1 Overview

An MQTT system consists of at least one server and clients communicating with the server. An MQTT server is commonly referred to as an MQTT message broker. All MQTT messages must go through the server, which means that clients cannot communicate directly with each other. The MQTT protocol requires the use of transmission protocols that provides ordered, lossless, bi-directional connections [18].

Data is transmitted using a publish-subscribe pattern, where senders publish a message which is then transmitted from the server to other clients called subscribers. Information sent is organized into topics where a publish message must contain a specific topic. When the server receives this message, it will forward it to any clients subscribed to the same topic.

An illustration of an MQTT server and clients is shown in fig. 3.4, which shows three clients connected to an MQTT server. When client1 publishes a message with the topic *temp/room1* it will be received by both client2 and client3. The subscription message from client2 contains a wildcard character, which means that client2 subscribes to all topics where the characters before the wildcard character match. For instance, if another client would publish a message with topic *temp/kitchen* client2 would receive this message while client3 would not.

3.5.2 Control packets

A description of all MQTT control packets is shown in table 3.3. The connect packet includes a keepalive value in seconds. If the client has not sent any control packet to the server for the duration of the keepalive, the server will disconnect the client. It is the client's responsibility to send a packet within the keepalive interval, and in the absence of any other control packet sent, it can send a ping request (PINGREQ) packet.

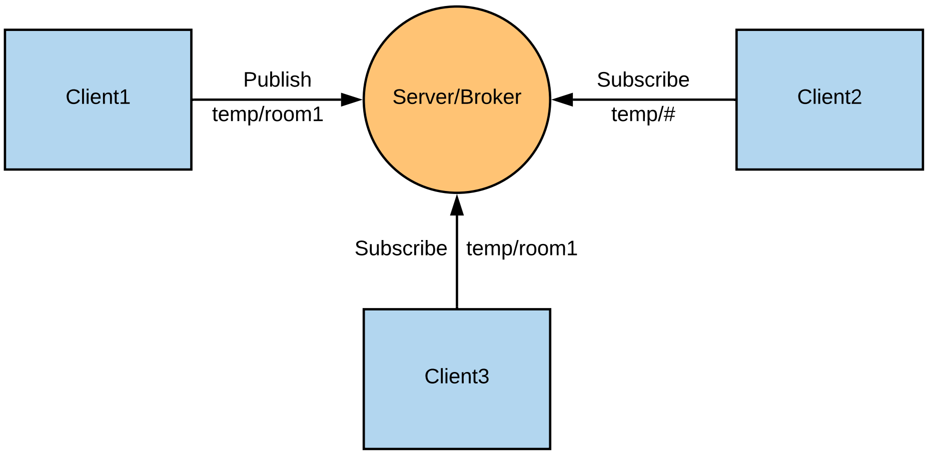


Figure 3.4: MQTT server and clients example.

Packet type	Direction	Description
CONNECT	Client to server	Client request connect
CONNACK	Server to client	Connect acknowledgment
PUBLISH	Both	Publish a message
PUBACK	Both	Publish acknowledgment
PUBREC	Both	Publish received(only for QoS2)
PUBREL	Both	Publish released(only for QoS2)
PUBCOMP	Both	Publish complete(only for QoS2)
SUBSCRIBE	Client to server	Client subscribe request
SUBACK	Server to client	Subscribe acknowledgment
UNSUBSCRIBE	Client to server	Unsubscribe request
UNSUBACK	Server to client	Unsubscribe acknowledgment
PINGREQ	Client to server	Ping request
PINGRESP	Server to client	Ping response
DISCONNECT	Client to server	Client disconnect from server

Table 3.3: Description of all MQTT control packets [18].

3.5.3 Quality of Service

MQTT has three different Quality of Service (QoS) levels that can be used with message delivering. The delivery is symmetric, which means that both the server and a client can take the role of a sender or receiver. The different delivery protocols in this section are used for the delivery of a message from a single sender to a single receiver. Consequently, when the server is forwarding messages to multiple clients, each client is treated as a separate independent transmission. The QoS levels are explained in table 3.4.

QoS level	**Description**
0	At most once delivery. Message is sent once and delivery is as reliable as the underlying transport layer protocol.
1	At least once delivery. Send the message multiple times until an acknowledgement is received.
2	Exactly once delivery.

Table 3.4: Description of the MQTT QoS levels.

Chapter 4

Specification and Design

This chapter contains a high-level overview of the complete system specification and design, including the embedded system and the cloud solution. Functional specifications are included for the embedded system, both hardware, and software and the cloud solution. A list of acceptance criteria based on the specification is presented for the complete system. Lastly, the design of the embedded hardware and data collection scheme is presented.

The hardware design presented is mostly based on the design and the results from the specialization project in [14]. The hardware shall be realized by designing and producing a custom Printed Circuit Board (PCB).

4.1 Overview

A high-level overview of the system is shown in the context diagram in fig. 4.1, where the main interactions between the different components are shown. The system is connected to an AMS meter and a cloud solution. The tasks of the cloud solution are data storage, monitoring, and possibly data analysis. The embedded system receives HDLC messages sent using the M-Bus physical layer standard from the AMS meter. The messages transferred between the system and the cloud are MQTT messages sent using either the TCP or TLS protocols.

The MQTT protocol should be used as the application layer protocol for communication between the system and the cloud solution. This choice is based on the low complexity and resource requirements of the protocol as shown in section 2.3, and more importantly since almost all popular IoT cloud platform services support or require the usage of MQTT. The desired transport layer protocol is the TLS protocol, which provides both encryption, authentication, and reliable transmission [13]. Real-time data concerning power usage is

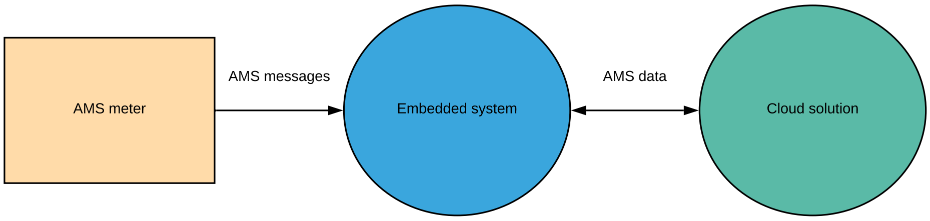


Figure 4.1: High-level overview of the complete system.

considered sensitive data and should be encrypted. Consequently, the TLS protocol should be used for data transmission.

The functional specification for the complete system is summarized in the list below.

- Measurement data from the AMS meter is received and processed on the embedded system.
- The system must be able to send and receive MQTT messages to a cloud solution.
- The TLS protocol is supported.

4.2 Specification

4.2.1 Embedded hardware

The hardware must be able to receive and process messages sent using the physical layer of the M-Bus standard. This includes the conversion of the M-Bus voltages into appropriate voltage levels for the MCU chosen. Moreover, it should be able to detect a power loss, and subsequently, be able to store the most recent measurement values in non-volatile memory. Notably, the most recent voltage values are of interest when a power loss occurs. Consequently, the system must be able to have a certain amount of backup power that can supply the MCU with power while the MCU stores the most recent meter values after a power loss.

The embedded system hardware should also be able to interact with several different peripheral units, including a modem/Wi-Fi module. It follows that the PCB to be designed should be designed as more of a development board than as a finished product. Additionally, by request of the thesis supervisor, the system should also be able to interact with analog signals, which means that ADC capabilities are required.

The HAN port of the AMS meters is capable of delivering a small amount of power. Considering that the PCB to be developed should be more of a development board than a finished product, the power supply should be made external. As a result, the board should not require an AMS meter for power, and it can be used for other applications as well.

The hardware specification is summarized in the list below.

- Convert M-Bus voltages down to appropriate levels.
- Must be able to detect a power loss.
- Contain a backup power source for use when a power loss occurs.
- Have some form of non-volatile storage used to store values after a power loss.
- Differential ADC capabilities.
- Enough peripherals to facilitate easy debugging and connection to other devices/-modules, including 2-3UARTs, SPI and I2C.

4.2.2 Embedded software

The software must be able to process the HDLC messages sent from the AMS meter, and subsequently extract the measurement values from the messages. A CRC calculation as described in section 3.3.1 should be performed on the message to verify the correctness of the content. Additionally, the software must be able to handle two-way communication with the cloud service using the MQTT protocol, which includes sending measurement values and receiving control messages from the cloud service. The software must be able to reconnect to the cloud in the case of a network failure or disconnect.

As sending all measurement values received from the AMS meters would result in a considerable amount of MQTT messages and measurement data sent, the software should only keep track of average, minimum, and maximum values within some specified time interval.

The software must also be able to handle a power failure, and subsequently, store the most recent measurement values in non-volatile memory. On the next startup, the software should check if there are any measurement values written to non-volatile memory because of a power failure. If there are any values present, these values must be published to the cloud service before the system can continue with regular operation.

In addition to the values included in both List 1 and List 2 in Appendix A, there are four energy usage values sent out every hour which is included in List 3. These values contain cumulative hourly import and export energy. As these values are only sent out every hour, a separate MQTT message containing these values should be published to the cloud service when they are received from the AMS meter.

A summary of the software specification is presented in the list below.

- Process, extract, and verify measurement values from AMS messages.
- Should work with different types of AMS meters.
- Must be able to reliably send and receive messages to the cloud solution using the MQTT protocol.

- Keeps track of average, minimum, and maximum values for every measurement value.
- Should reconnect to the cloud if disconnected.
- Handle a power fail by storing the most recent measurement values in non-volatile storage.
- Detect a previous power fail at startup and send the power fail data to the cloud.
- Send hourly energy measurement data to the cloud.

4.2.3 Cloud solution

The cloud solution must be able to handle the reception of data from numerous clients at a time using the MQTT protocol. Additionally, it should have some features for monitoring and data visualization. It should be reliable with minimal downtime.

Another important factor for choosing a cloud solution is pricing. Ideally, a free IoT cloud service should be used in the thesis work. Furthermore, the cloud solution should be easy to use and set up in order to avoid spending too much time on getting the cloud service to work.

4.3 Acceptance criteria

Based on the functional specifications presented earlier in this chapter, a list of acceptance criteria for the complete system can be made. An implementation of the system can be considered successful if all the acceptance criteria presented in table 4.1 are satisfied. Throughout the thesis, the acceptance criteria are referred to by their label in table 4.1.

Label	Acceptance criteria
AC1	The system is able to work with different AMS meter types. That is both three-phase meters and single-phase meters from different suppliers.
AC2	AMS HDLC messages are received by the embedded system and the measurement values are extracted from these messages if the message checksum calculation is correct.
AC3	The system can be connected to a cloud service using the MQTT application layer protocol together with the TLS protocol.
AC4	Using the MQTT protocol with support for QoS1 message delivery is possible.
AC5	The system correctly calculates average, minimum and maximum values for all meter measurements within some time interval.
AC6	Two-way communication with the cloud is supported, where the measurement sending frequency can be adjusted by sending a message to all clients from the cloud service.
AC7	In the case of connection/network loss, the system continues to receive AMS measurement values.
AC8	In the case of connection/network loss, the system will reconnect if possible.
AC9	Power failures are detected and the most recent voltage values are stored in non-volatile memory. The system must have enough backup power to function properly until these values have been stored.
AC10	After a power failure, the voltage values written to non-volatile memory are sent the the cloud solution before the system continues operating normally.
AC11	The hourly cumulative energy values in List 3 shown in Appendix A are sent to the cloud.
AC12	The embedded systems contains extra peripheral interfaces including UART, SPI, I2C and differential ADC capabilities.

Table 4.1: List of acceptance criteria for the complete system.

4.4 PCB design

A high-level overview of the PCB design is shown in fig. 4.2. The block containing the voltage regulator and backup power is responsible for regulating the input from the power supply and containing some backup power in case of a power outage. The M-Bus to TTL block must convert the M-Bus voltage, and it must also be able to detect a power loss on the M-Bus lines. The board should also have some status LEDs for state indication, and headers for connecting to various peripheral units. This is included in the block containing IO and LEDs.

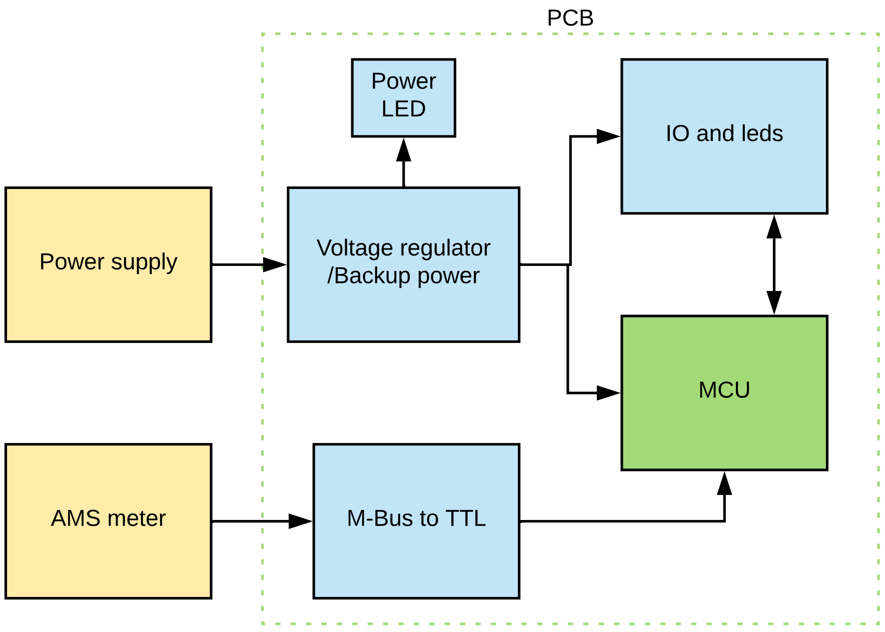


Figure 4.2: High-level design of PCB.

4.5 Design of data collection scheme

The system should send average, minimum, and maximum values within a time interval as specified earlier in this chapter. The messages from the embedded system to the cloud should contain the average, minimum, and maximum for every measurement value the meter sends out. A list of these values is included in Appendix A. One cycle of this process for the embedded system is illustrated in fig. 4.3, where the *Should send* choice checks if the sending period has been reached and if it is time to send the data. After being received by the cloud service, the measurement values should be stored for later analysis.

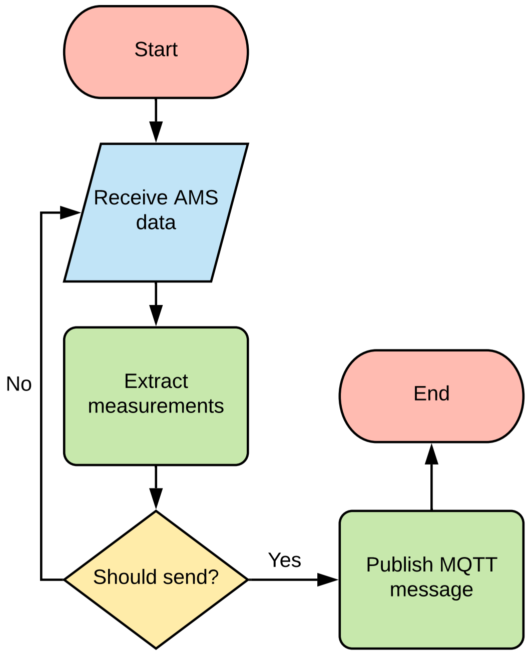


Figure 4.3: Flowchart illustrating one cycle of data collection for the embedded system.

Chapter 5

Implementation

This chapter contains details on how two embedded systems have been designed and implemented. One of the systems was used for data collection from multiple sources, while the other system serves as an alternative implementation with different features. The primary system implemented and used for data collection contains an ATmega324PB as the MCU. The alternative implementation contains a much more powerful ESP32 chip which has built-in Wi-Fi capabilities. Throughout the rest of the thesis, the system containing the ATmega324PB is referred to as the ATmega system, while the system containing the ESP32 is referred to as the ESP32 system.

Altium Designer has been used to create the schematics and designing the boards presented in this chapter. Both PCB designs have been implemented as 2-layer PCBs where the bottom layer has been used as a ground plane. Most of the components chosen are surface mount, resulting in smaller PCBs and fewer drill holes.

5.1 Hardware ATmega system

The complete schematic for this PCB design is included in Appendix B1, and the corresponding list of components is included in Appendix B2. This design is based on the work done in [14] with some additions and improvements, for instance, removing dead copper on the board, changing the power circuit, adding ADC inputs, and improving the routing of the PCB.

5.1.1 Components

This section presents an overview of the components chosen for the PCB design shown in Appendix B1. An 11.0592MHz crystal oscillator was chosen as the clock source for the MCU. This frequency was chosen such that the system can easily work with several different standard baud rate values as this frequency is a multiple of these baud rates. The three LEDs on the PCB are low-current LEDs which only draws a current of 2mA each. LEDs D1 and D2 are intended as status LEDs, which can be used by the embedded software to signal or show the state of the system.

MCU - ATmega324PB

The ATmega324PB was chosen as the MCU for the system as it was in [14]. It is a high performance 8-bit RISC-based MCU from the AVR family of microcontrollers with 32 KiB flash program memory, 2KiB SRAM, and 1KiB EEPROM. It includes plenty of digital communication peripherals, for instance, three hardware UART modules, I2C, SPI, JTAG, and differential ADC. Based on the requirements of the embedded system, the ATmega324PB should be well suited for implementing the data acquisition software. Furthermore, all the peripherals result in multiple debugging and connection possibilities.

The AVR microcontroller architecture is based on a modified Harvard architecture, which means that instruction memory and data memory are separate. An illustration of this architecture is shown in fig. 5.1, which shows that flash memory and SRAM memory are physically separated. The AVR architecture allows for storage of constant values in flash memory (program memory) which is much larger than the SRAM data memory. This feature is very useful for storing look-up tables and strings, as they will not take up any of the SRAM space when stored in program memory. Values stored in the program memory must be received through special macros defined in the *avr-libc* library. The software implementation in section 5.2 has thoroughly placed all strings and constant tables in program memory to avoid unnecessary usage of SRAM.

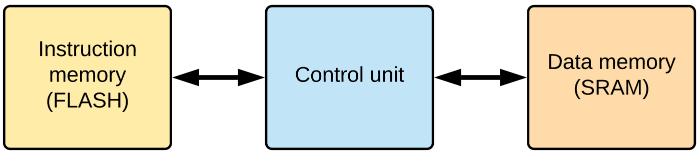


Figure 5.1: Harvard computer architecture used in AVR MCUs.

M-Bus transceiver - TSS721A

The TSS721A is an M-Bus transceiver IC developed by Texas Instruments in accordance with the M-Bus specification. The most important feature of this IC is the conversion of the M-Bus voltages down to a level which is acceptable for the MCU, that is it works as a level shifter. In addition to voltage conversion, it has a built-in power failure functionality that triggers when there is a loss of voltage on the M-Bus lines connected to it, which will pull one of the pins of the chip low. Additionally, it has an integrated LDO regulator which can be used to power a low power MCU.

LDO regulator - LM3940

The voltage regulator chosen is an LDO regulator specifically designed for 5V to 3.3V conversion, and it can output 1A. The typical dropout voltage of the regulator is 0.5V, which means it will be able to keep an output voltage of 3.3V with an input voltage as low as 3.8V.

Wi-Fi module - ESP8266

In order to be able to upload data to the cloud solution, a module containing an ESP8266 chip was chosen. It is a low-cost Wi-Fi module which has support for the TLS 1.2 protocol. It can be interfaced using AT commands in order to connect and transfer data. The module chosen for this project is an ESP-01 module which has 1MiB built-in flash memory and the ESP8266 chip. An SDK is available from *espressif* which is the producers of the ESP8266 chip. The SDK adds support for flashing custom software to the chip, or modification of the default software running on the chip.

5.1.2 Power circuit

The schematic for the power circuit is shown in fig. 5.2. The power connector chosen is a micro-USB connector which should be supplied from a 5V power supply. Both D5 and D4 are Schottky diodes, where the role of D5 is to prevent the backup power stored in capacitor C13 to drive the VBUS line backward when the power supply is disconnected. The D6 diode is a transient-voltage-suppression (TVS) diode used for ESD protection. The purpose of the resistor R10 is to limit the amount of current used for charging C13. Schottky diode D4 provides a low-impedance path from C13 to the input of the LDO in the case of power failure. Both C11 and C12 are capacitors chosen from the information in the LDO regulator datasheet. Lastly, the power circuit contains the low-current power LED D3, which should emit light whenever the circuit has power.

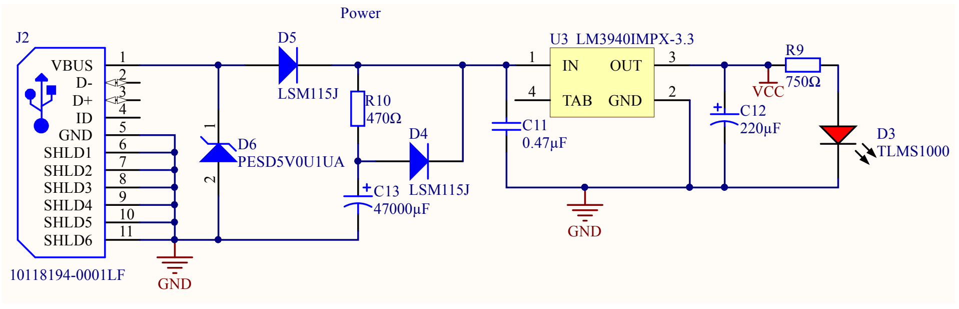


Figure 5.2: Power circuit schematic for the ATmega PCB.

The power circuit includes backup power stored in the supercapacitor C13 such that AC9 in section 4.3 can be satisfied. Alternatively, a small battery could have been used as backup power. However, in the case of a power failure, there is no need to keep the MCU running after storing voltage measurement values in non-volatile memory, as there is nothing meaningful the MCU can keep doing. The assumption that a power failure means that the AMS meter will temporarily stop sending data is made. Consequently, the MCU only needs power for the amount of time it will take to store the most recent voltage values in MCU EEPROM.

The supercapacitor is located before the LDO regulator as opposed to after for several reasons. The voltage difference tolerated is higher at the regulator input than output. With the chosen crystal oscillator the MCU will be able to operate until the voltage drops below approximately 2.7V [15], which gives a voltage difference on the output of 3.3V-2.7V=0.6V. With a regulator dropout voltage of 0.5V, the voltage difference on the input is approximately 5V-3.8V=1.2V, which means that more energy can be stored in the capacitor by placing it before the regulator. Additionally, placing the capacitor before allows the regulator to operate normally until the voltage drops below the dropout voltage.

It is important that the supercapacitor can store enough power for the time it will take to store the voltage values in the EEPROM of the ATmega. The current-voltage relation for an ideal capacitor is given in eq. (5.1) and eq. (5.2). Further, if the current is constant and we set $t_0 = 0$, the voltage-current relation is linear and given in eq. (5.3). With the component values shown in fig. 5.2 and if the part of the circuit after the supercapacitor is ignored, the time constant of the RC circuit is given by $\tau = RC = 470\Omega \cdot 47000\mu F = 22.09s$. This number is not accurate since the LDO connected after the capacitor will affect the electrical characteristics of the circuit when it is activated. However, based on the calculated time constant, the capacitor should be able to reach its maximum energy stored in a reasonable amount of time.

$$I(t) = C\frac{dV(t)}{dt} \tag{5.1}$$

$$V(t) = \frac{1}{C}\int_{t_0}^{t} I(\tau)d\tau + V_0 \tag{5.2}$$

$$V(t) = \frac{I}{C}t + V_0 \tag{5.3}$$

The Schottky diode D5 has a small forward voltage drop which varies with the amount of current passing through. Under normal operation conditions at 25°C the voltage drop will be ranging from approximately 0.1V to 0.2V according to the datasheet. To calculate the discharge time for the supercapacitor, the V_0 voltage is assumed to be 4.8V, and a constant current draw of 50mA is assumed. This high of a current draw is a high estimate as the ATmega324PB draws approximately 5-6mA when active with an 11.0592MHz crystal oscillator [15], the LEDs draw 2mA each, and the voltage regulator will also draw some current. Using eq. (5.3) the discharge time is calculated to:

$$t = \frac{C}{I}(V - V_0) = \frac{47000\mu F}{-50mA}(3.8V - 4.8V) = 0.94s$$

By setting the EEPROM write mode of the ATmega to write only, the time to write a single byte to EEPROM is approximately 1.8ms [15]. Hence, in 0.94s, approximately 522 bytes can be written to EEPROM, which should be more than enough to store multiple measurement values.

5.1.3 ADC circuit

The possibility of connecting analog signals to the board has been added by request of the thesis supervisor for possible use in future projects. The schematic is shown in fig. 5.3 where the resistors R11-R14 are protection resistors connected to the differential ADC input pins of the ATmega. As the ATmega can only handle positive voltages for the ADC inputs, the resistors R15-R20 serve as voltage dividers. Hence, differential ADC signals with one negative input can be handled by choosing appropriate resistor values. If there is no need for differential ADC channels, resistors R15-R16 can be omitted, resulting in four single channels available for the ADC. Input signals are to be connected to the header P5 which also has connections to ground. The header P6 shown in Appendix B1 is for connecting an external ADC reference voltage if this is required.

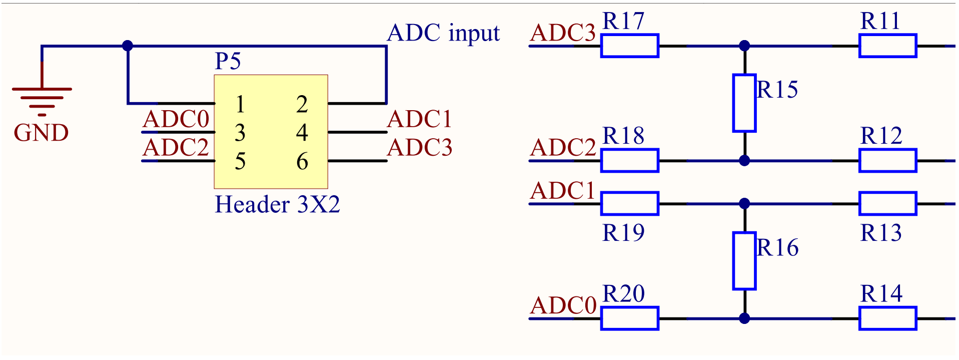


Figure 5.3: ADC circuit for the ATmega PCB.

5.1.4 M-Bus conversion circuit

The schematic for the conversion of M-Bus signals is shown in fig. 5.4. This part of the design is the same as the one used in [14]. The values of resistors and capacitors are chosen according to the values specified in the datasheet of the TSS721A IC. The BAT pin on the TSS721A is used to adjust the voltage level of the conversion, and it is connected to VCC (3.3V). As a result, the M-Bus signals are converted down to 3.3V and sent out on the TSS721A TX pin. The PF pin is the output of the power failure functionality and is directly connected to one of the hardware interrupt pins of the ATmega.

Both the TX and RX signals are connected to one of the hardware UARTs of the ATmega. It is not necessary to connect the RX pin of the TSS721A, as the AMS meters only outputs

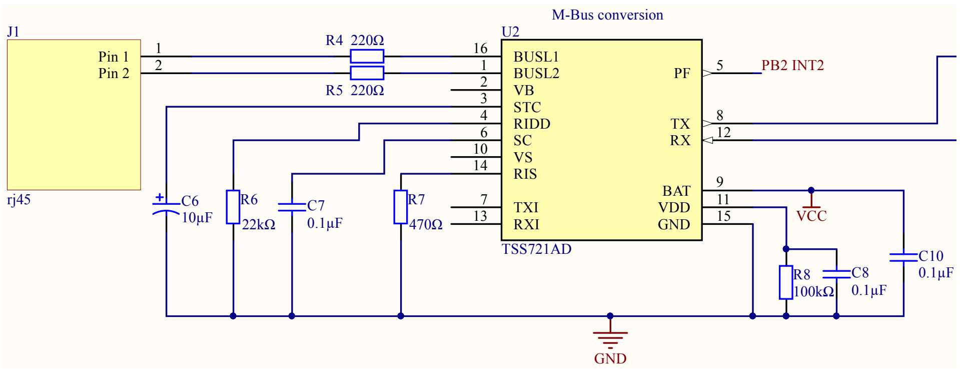


Figure 5.4: M-Bus conversion schematic.

data and will not accept any messages from M-Bus slaves. It is worth noting that resistors R4 and R5 are rated for 0.5W because of the high voltage on the M-Bus lines. Lastly, the output from the internal LDO on the TSS721A IC is connected to ground through the resistor R8 as power from the M-Bus lines is not utilized.

5.1.5 Headers and peripherals

Multiple headers are included in the design of the PCB shown in Appendix B2. Header P1 contains general IO pins, I2C, and SPI. P3 is a power header with VCC and GND, and P4 is a JTAG header for programming and debugging. The header P2 has been designed such that the ESP-01 Wi-Fi module can be inserted directly without the use of any extra wires. Moreover, P2 contains UART and I2C peripherals.

5.1.6 PCB result

A prototype of the PCB was ordered from *JLCPCB* in China, and soldering and testing of the board were performed at NTNU. After the prototype was confirmed working, a batch of PCBs with all components assembled was ordered from Chinese company *Elecrow*. The result is shown in fig. 5.5 which shows the complete system used for data collection in chapter 6. The dimension of the board is 68.5mm x 44.3mm. The top layer contains traces and a power plane, while the bottom layer is a ground plane with only a few short traces.

5.2 Software ATmega system

All the software for ATmega324PB has been developed from the ground up without the use of any external libraries or operating system. This approach results in a complete

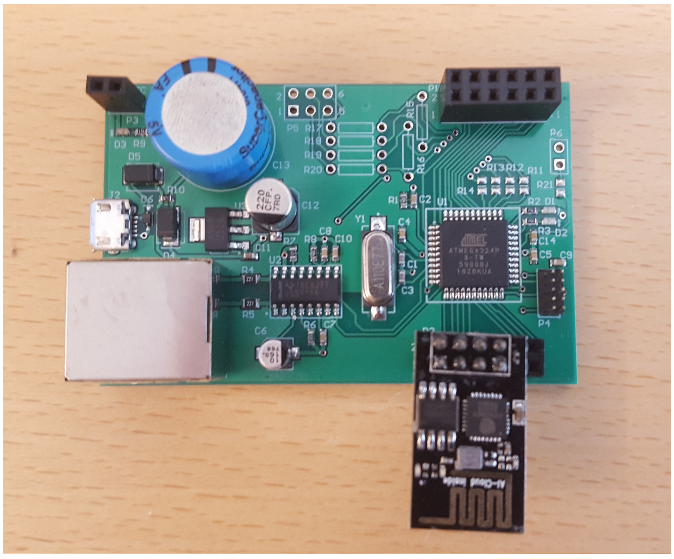


Figure 5.5: Assembled ATmega PCB with an ESP-01 Wi-Fi module connected.

understanding of every part of the software and the interactions between software modules. This section contains an overview of the different software modules and the complete embedded software. The software modules described in section 5.2.4 and section 5.2.5 are based on work performed in [14] with significant changes and redesigns. The main program operations is described in section 5.2.8.

The software is written mostly in C with a few inline assembly functions. *AVR-GCC* is the toolchain used for compiling and the *avrdude* tool has been used to flash the program and set fuse bits using an *Atmel-ICE* programmer. The software has only been tested with the ESP-01 Wi-Fi module. However, it should not be too complicated to add support for other modules as long as they support using AT commands.

Documentation for the code has been added in the form of *Doxygen* comments in the source code. The use of *Doxygen* allows documentation to be generated in an HTML or PDF file, which can be used to find documentation for all the different module interfaces.

5.2.1 Interrupts and atomic access

The 8-bit architecture of the ATmega324PB means that only operations on 8-bit variables are guaranteed to be executed atomically. Consequently, any variable larger than 8-bit, which is modified from an interrupt service routine (ISR), requires interrupts to be disabled while the program is accessing the variable to guarantee atomic access. This has been implemented by using the macros for creating atomic blocks defined in *util/atomic.h* from the *avr-libc* library.

5.2.2 Timer

The embedded software must be able to handle IO-data from both the AMS meter and the Wi-Fi module. Therefore, it is useful to have a method of keeping track of time. An interrupt driven timer which ticks approximately every millisecond has been implemented using the 8-bit timer/counter module of the ATmega. This module contains a *Compare Match Output Unit* which can be used to generate an interrupt when an 8-bit counter reaches a specific value [15]. With a crystal oscillator frequency of 11.0592MHz and the timer prescaler set to 64, the match value of the counter to get one tick per millisecond is calculated below.

$$\frac{0.001s}{\frac{64}{11059200Hz}} = 172.8 \approx 173$$

As the compare match register must be set to an 8-bit value, the timer interrupt will activate marginally slower than every millisecond when the compare match is set to 173. This issue is accounted for by adding an extra millisecond to the timer variable every 864 time the interrupt activates, which is precisely when the extra time has added up to one millisecond. The timer module stores the current timer value in an unsigned 32-bit integer value, which can be accessed by calling a function called *timer_ms* which returns this value. Additionally, a delay function using this timer has been implemented, which will delay for a certain amount of milliseconds.

5.2.3 Watchdog timer

The ATmega324PB contains a watchdog timer which uses an internal separate 128kHz oscillator. The watchdog has three different operating modes, interrupt, system reset, and combined interrupt and system reset mode. In interrupt mode, an interrupt is activated when the counter reaches a time-out value. In system reset mode, the watchdog resets the system when the timer expires. The third mode is a combination of the two other modes.

The maximum value the watchdog timer can be set to will generate a watchdog time-out approximately every 8 second. In order to increase the amount of time before a system reset, the watchdog timer is first activated in interrupt mode. An Interrupt Service Routine (ISR) which increases a counter is run when the watchdog times-out. If the counter reaches a set value, the mode of the watchdog timer is switched to system reset mode, and the next time-out will result in a system reset. Consequently, the watchdog reset timer can be set to a multiple of 8 seconds if this is desirable.

The primary usage of the watchdog timer is to reset the system after a power failure happens. Usually a power failure should reset the MCU when the brown-out detection detects a too low voltage value, however, if the cause of the power failure interrupt is not an actual power failure, the watchdog is used to reset the MCU. Additionally, it serves as a safety mechanism if the software gets stuck in an unknown state or infinite loop.

5.2.4 Drivers

EEPROM

A simple driver for writing and reading bytes to EEPROM has been implemented. Since the EEPROM has a limited write/erase cycles of 100,000, unnecessary writes should be avoided. Reading from EEPROM does not affect the life cycle of the EEPROM. Therefore, a function called *eeprom_update*, which first checks the current value in the EEPROM address to be written, has been implemented. If the current value is the same as the one to be written, the write will not be performed, thus avoiding unnecessary writes.

LEDs

The driver for the status LEDs consists of simple macros for initialization, toggle, set, and clear functionality. These macros are used to control the two status LEDs on the board, one yellow and one orange.

UART

The ATmega324PB contains three hardware UART modules. The UART0 is connected to the TSS721A chip, while UART1 and UART2 are available from the header P2 on the PCB. The implementation uses UART1 for communication with the Wi-Fi module, while UART2 is used for printing debugging information. Interrupt driven receive functionality has been implemented for both UART0 and UART1 inspired by an existing library [11]. The receive interrupt generated when a byte is received will put the byte into a circular buffer. The module interface includes functions for getting available bytes in the buffer, flushing the buffer, extracting the byte at the tail end of the buffer, and functions for sending strings and data through the UART.

AT commands

The AT command set (Hayes command set) is a specific command language originally developed to interact with modems. The ESP8266 chip used in this system is controlled by using AT commands. A module for sending AT commands and receiving responses through UART1 has been implemented. The commands to be sent are strings which are required to be located in the program memory of the ATmega. Functionality for waiting for a specific response has also been implemented. Consequently, the result of an AT command can be confirmed by waiting for a specific response from the Wi-Fi module. The most common response expected is *OK*, which indicates that the AT command sent was executed successfully by the module. Alternatively, an error message is received from the module.

5.2.5 AMS message reception

The AMS messages are received in HDLC frames and are extracted from the UART0 buffer. A function which reads in a complete HDLC frame into a buffer and performs a checksum calculation as explained in section 3.3.1, has been implemented. The function has been implemented with a time-out parameter, which means it will return an error code after the duration of the time-out has passed without receiving a message.

The procedure of reading a message will return a value to indicate if the reception was successful or not. Reception of a message is successful only if a whole message is received and the calculated checksum is correct, which results in the value 0 being returned.

An initialization procedure is required for this module to be able to handle messages from multiple different meter types. The initialization will try to read in HDLC messages until a message with known length is successfully received. The length of the received message is used to identify the meter type and used to initialize an array containing offset values. Subsequently, the offset values are used for measurement value extraction from the HDLC frames received. Every AMS message contains a timestamp value which is converted into a UNIX time value.

5.2.6 Power failure

When the power failure interrupt of the TSS721A IC is activated, the most recent voltage measurement values are stored in the EEPROM of the ATmega. The corresponding ISR will loop through all the voltage values in a buffer containing recent voltage values, and write them to EEPROM. After all values have been written, the system will enter an infinite loop which will cause the system to reset because of the watchdog timer if not reset by brown-out detection.

The scheme used for storing the data is shown in fig. 5.6 where the size of each field in bytes is indicated. The flag byte contains two flags, one to indicate if the meter is a three-phase meter and one to indicate that a power failure has happened. In that way, at system startup, only the flag byte has to be read to check if a power failure has occurred. The timestamp values are 32-bit UNIX time values, and each voltage value is a 16-bit value. Three-phase meters have three voltage values, while single-phase meters only have one. Consequently, the three-phase flag in the flag byte indicates how many voltage values are stored for each timestamp. The total number of voltage measurement values to be stored at a power failure is configurable in the software.

The CRC checksum is calculated using the same method as in section 3.3.1. The checksum is used to validate the correctness of the measurement data stored in EEPROM, which is useful if the system was in the middle of writing values to EEPROM when voltage brown-out detection activated. A measurement is ignored if the CRC calculation fails when reading the measurement data from EEPROM. Each 16-bit CRC value is associated with a single timestamp and the voltage values for this timestamp.

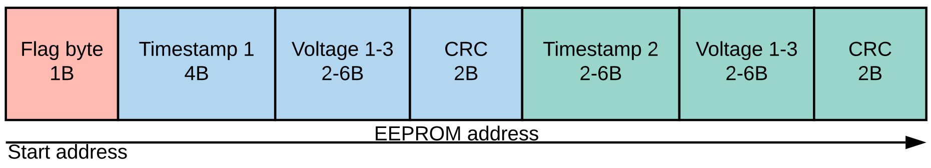


Figure 5.6: Storage of voltage measurement values in EEPROM.

At startup, the system will check for a power failure by extracting the first byte at the start address for power fail data from EEPROM. This process is illustrated in a flowchart in fig. 5.7. The process is blocking, which means that if a power failure happens, the system will continue to try connecting to the MQTT server and uploading the data until it succeeds. After the data has been successfully sent, the EEPROM will be cleared, and the EEPROM write mode is set to write only. This halves the writing time of a byte to EEPROM as the system does not have to perform an erase before a write [15]. After the power failure process, the system will enter the normal initialization mode explained in section 5.2.8.

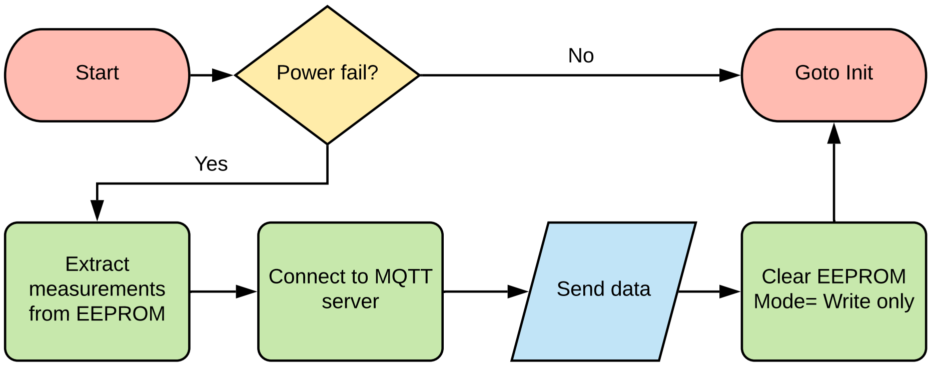

Start
Power fail?
No
Goto Init
Yes
Extract measurements from EEPROM
Connect to MQTT server
Send data
Clear EEPROM Mode= Write only

Figure 5.7: Flowchart showing the power failure check at startup.

5.2.7 MQTT implementation

An MQTT client for communication and transfer of data to the cloud solution has been implemented. The client supports subscribing and publishing messages using either QoS0 or QoS1. Two buffers are used in the implementation; one send buffer and one receive buffer. No queuing features are implemented. Hence, the client supports at most one unconfirmed outgoing message when publishing using QoS1. When using the implemented client, one should connect to the MQTT server and subscribe to topics before starting to publish messages.

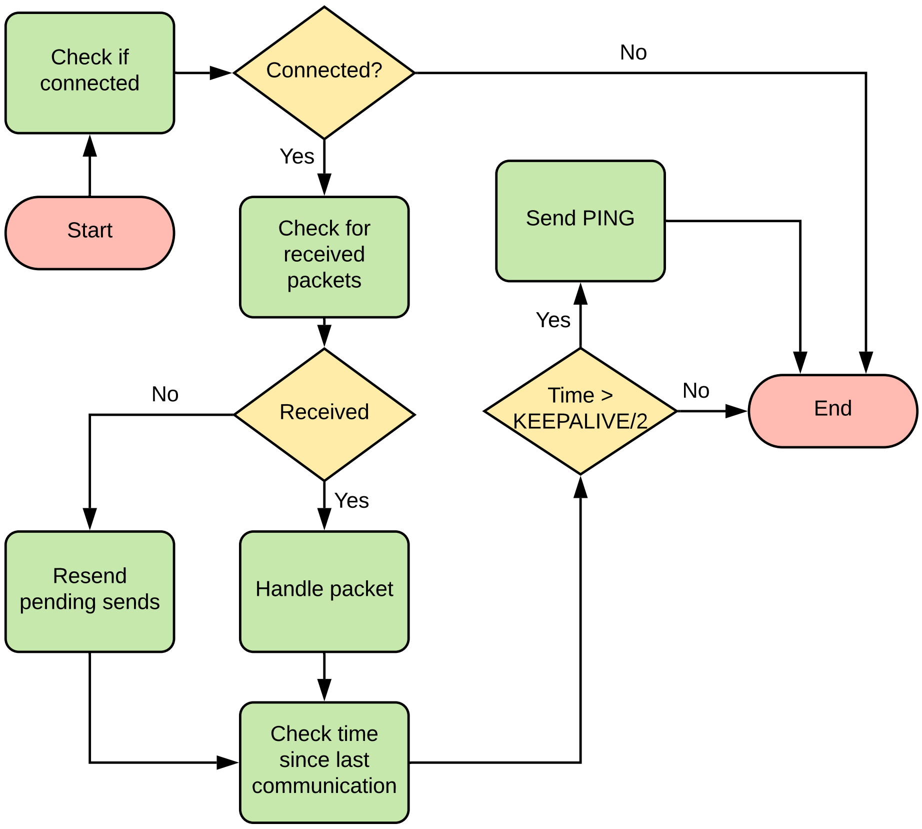


Figure 5.8: Flowchart illustrating one iteration of the *mqtt_loop* function.

A function called *mqtt_loop* has been implemented to easily integrate MQTT functionality into the main program. This function should be called periodically to check for received packets and maintain the MQTT connection. The tasks of this function are shown in a flowchart in fig. 5.8. First, the connection will be checked by comparing the time after the last message sent/received and the MQTT keepalive value. If the client has timed out, the function will return a value to indicate that a disconnect happened. Multiple actions can be executed in the handle packet step in fig. 5.8 depending on the packet type received. An explanation for each packet action is included in table 5.1. Finally, the function checks the time passed since the last communication with the server. If this time in seconds exceeds half the value of the keepalive time, an MQTT PINGREQ packet is sent to avoid getting disconnected.

Packet	**Action**
PUBLISH	If the client has subscribed with QoS1, a publish acknowledgment message is sent back. The payload received is written into a receive buffer which must be provided by the function caller.
PUBACK	Clear a pending send flag since the message to be sent has been acknowledged by the server.
PINGRESP	No action needed other than record the time of the received packet.

Table 5.1: Software actions corresponding to the MQTT packet received.

5.2.8 Main program operation

A state diagram showing the states of the main program is shown in fig. 5.9. After initialization is complete, there are only two states the system can be in, the *Connected* state and *Not connected* state. Although the power failure process presented in fig. 5.7 can be considered a state, it is not included in the state diagram as there are not any software transitions to this state. The state diagram explains the usage of the status LEDs, where the orange LED shows if the AMS message initialization process has been completed, while the yellow one shows if the system is connected to the cloud.

The *Init* state is reached after initializing all drivers and peripherals and after the power failure process in section 5.2.6. In this state, the system will continue to try initializing the AMS message reception module. There is no need for a transition back to the *Init* state, as a stop of messages from the AMS meter implies that a power failure has happened, or the cable to the meter has been pulled out. Both of these events would result in a power failure interrupt causing a system reset.

In the *Not connected* state the system will continue to receive AMS messages while it tries to connect to the cloud. This is illustrated in fig. 5.10, which shows one iteration of the program in this state. The *Handle msg* step consists of updating the minimum and maximum value and adding to a cumulative variable for each of the measurement

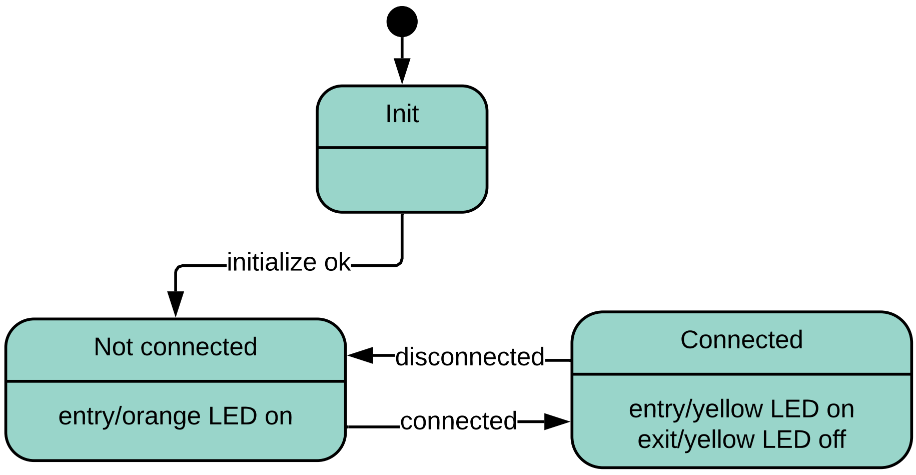


Figure 5.9: UML state diagram for the embedded software.

values present in the message. The minimum, maximum, and cumulative values are stored in a buffer which also holds the timestamp of the latest received message. Only when the system is connected and manages to subscribe to the MQTT topic *freq* for receiving changes to the sending frequency, will the program transition to the *Connected state*.

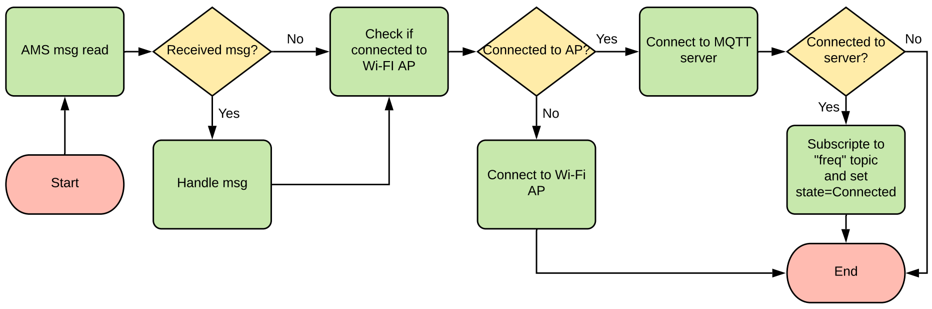


Figure 5.10: Flowchart showing one iteration in the *Not connected* state.

Normal system operation in the *Connected* state is shown in fig. 5.11. The process of receiving and extracting measurement values is the same as in the *Not connected state*. Following the handling of a possible new AMS message, is a check if the system should send values to the cloud. This check compares the number of AMS messages received since the last send to the number of AMS messages expected within the current send period, where the default send frequency is set to once every minute. If the system should send the values to the cloud, it will compute the average values for the last period and send the average, minimum and maximum values to the cloud together with the timestamp value for the most recent AMS message received. All data is sent as raw byte data, which means the receiver must decode the data in the payload. The *MQTT loop* block performs the functionality shown in section 5.2.7. After running the *mqtt_loop* function, the software checks if an MQTT message was received. If an MQTT message is received on the *freq* topic, the software will change the data sending frequency to the value specified in the message payload. The software only accepts a sending frequency of once every 1-60 minutes.

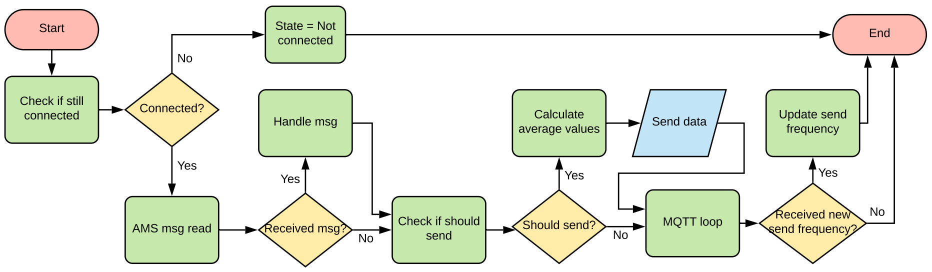


Figure 5.11: Flowchart showing one iteration in the *Connected* state.

5.3 Hardware ESP32 system

The schematic for an alternative hardware implementation for the embedded system is shown in Appendix C1, and the corresponding component list is included in Appendix C2. This implementation purposely does not satisfy all the acceptance criteria set in section 4.3. It serves as an alternative implementation containing a more powerful processing unit and made exclusively for connecting to the cloud using Wi-Fi. The processing unit chosen is an ESP32-WROOM-32D module from *espressif* which contains a powerful dual-core CPU and is built for use with Bluetooth and Wi-Fi applications. It is the same processing unit as used in the module presented in section 2.2.

It is worth noting that this module requires a time delay of minimum 50μs in between when 3.3V is applied to the chip and when the enable pin EN is brought high [7]. This is accomplished with the capacitor C3 shown in Appendix C1. A tactile push-button is connected to the EN pin to allow easy resetting of the module, while another button is connected between IO0 and ground that must be pushed in at startup to enter programming mode.

5.3.1 Components

The choice of components is very similar to the design in section 5.1. The TSS721A chip is still used together with a micro-USB and an RJ45 connector. The LDO 5V-3.3V regulator was changed to a cheaper module than the one used in section 5.1. A power LED, and two status LEDs are included as in section 5.1. Lastly, IO headers connected to the IO pins of the ESP32-WROOM-32D are included together with a power header, programming header, and JTAG header.

MCU - ESP32

The ESP32-WROOM-32D module contains an ESP32 chip which designed for use with 2.4GHz Wi-Fi, Bluetooth and Bluetooth Low Energy (BLE). Moreover, it integrates a rich set of peripherals and sensors, for instance, UART, I2C, SPI, Ethernet, and hall sensors [7]. A list of some of the features of the ESP32-WROOM-32D module is presented below.

- 4MiB flash program memory
- 520KiB SRAM
- Dual core Xtensa 32-bit LX6 microprocessors
- 802.11 b/g/n 2.4GHz Wi-Fi support
- Bluetooth and BLE support
- Ultra low power (ULP) co-processor

5.3.2 PCB result

A PCB prototype was ordered from *JLCPB*, and soldering was performed at NTNU. The result is shown in fig. 5.12. The dimension of the board is 63.8mm x 32.9mm. The top layer contains traces and a wider power trace, while the bottom layer is a ground plane with only a few traces.

5.4 Software ESP32 system

The software for this system has been developed using the *ESP-IDF* framework, which is the official SDK for the ESP32 chip. The SDK uses FreeRTOS as the operating system, and it includes modules for several protocols, including an MQTT module which was used in this implementation. The module has been programmed by using the ESP32 toolchain and a *CP2102* USB to UART bridge to flash the program.

The message reading and decoding have been implemented as a FreeRTOS task illustrated in fig. 5.13. AMS messages are received, converted to the JSON format, and sent to the cloud. The MQTT module is running as a separate FreeRTOS task which supports buffering of multiple messages and reliable sending.

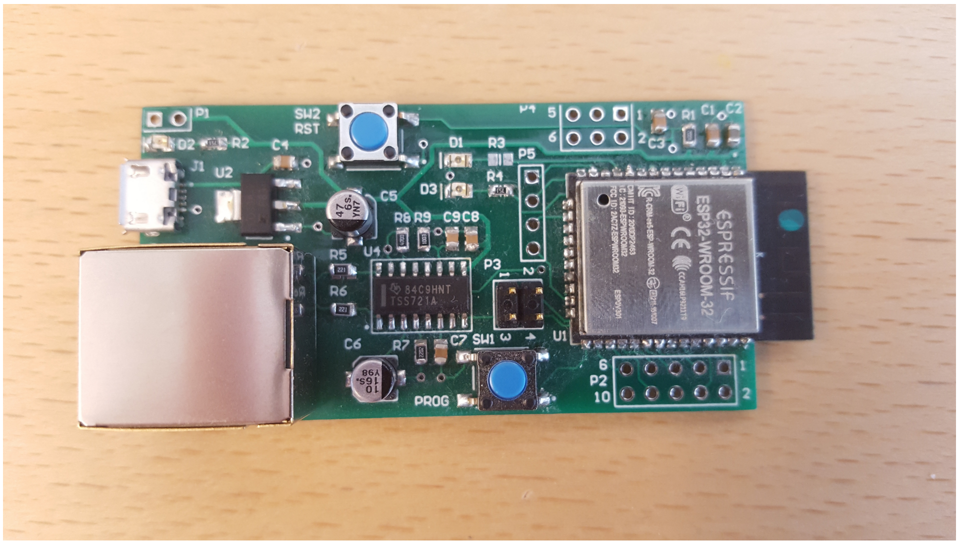


Figure 5.12: Assembled ESP32 PCB.

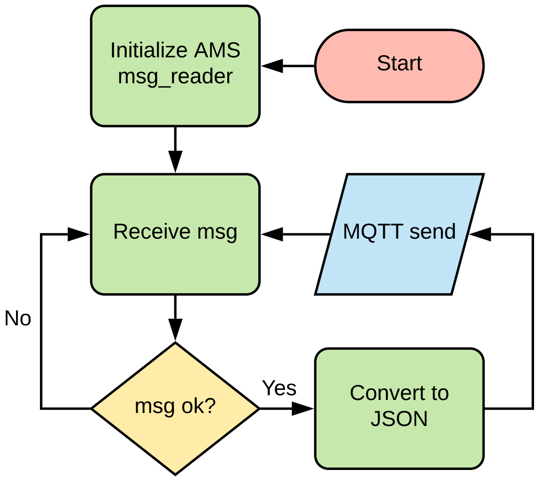


Figure 5.13: Flowchart showing the actions of the implemented FreeRTOS task.

This software implementation relies heavily on the SDK provided as opposed to the implementation in section 5.2. The AMS message reader module from section 5.2 was ported to the ESP32 and the *ESP-IDF* SDK.

5.5 Cloud solution

Amazon Web Services(AWS) IoT was used as the cloud solution for the data collection. The features available in AWS IoT Core and the AWS management console makes it easy to register new devices and monitor the activity of the clients. All clients are required to connect using TLS version 1.2 and have valid credentials which are received when registering a new device.

AWS IoT allows for easy integration with other AWS features, including cloud computing, data storage, analytics, machine learning, and more. This is achieved by creating an AWS *lambda rule* which can either send the MQTT message received directly to another AWS service or call a custom function, inputting in data from the MQTT message that triggered the rule.

5.6 Data collection

Initially, the entire data collection system was planned to be implemented using AWS services. However, only some of the services are available for free users, and there are additional limitations for free users. For instance, there is a limit to how many MQTT messages can be sent and how many data points can be stored in the storage service (Amazon S3). These limitations, together with the time which had to be spent to get familiar with the different services and implementation methods, resulted in AWS IoT only being used for monitoring and as the MQTT server.

The figure shown in fig. 5.14 shows the setup for the data collection. An additional client shown as the data collector is connected to the cloud. The data collector uses the MQTT topic wildcard character # to subscribe to all data received on the topics *avg/*, *hr/* and *pf/* corresponding to average data, hourly energy data and power failure data. As shown in the figure, all clients have been given a unique identifier added in the topic name(*id1* and *idn*). This identifier allows the data collector to recognize the sender a message.

The data collector was implemented in python, using the *paho-mqtt* python package to connect to the AWS IoT. The python script subscribes to the topics shown in fig. 5.14, receives MQTT messages, converts the raw data into strings and stores the strings in comma-separated values (CSV) files. A folder structure is created, where a folder is created for each client. Subfolders for average, hourly, and power fail data are created, where the CSV files are stored. Each file contains data for one day where the filename is set to the date of data reception. This approach to data collection requires a PC or server that can run the data collector software.

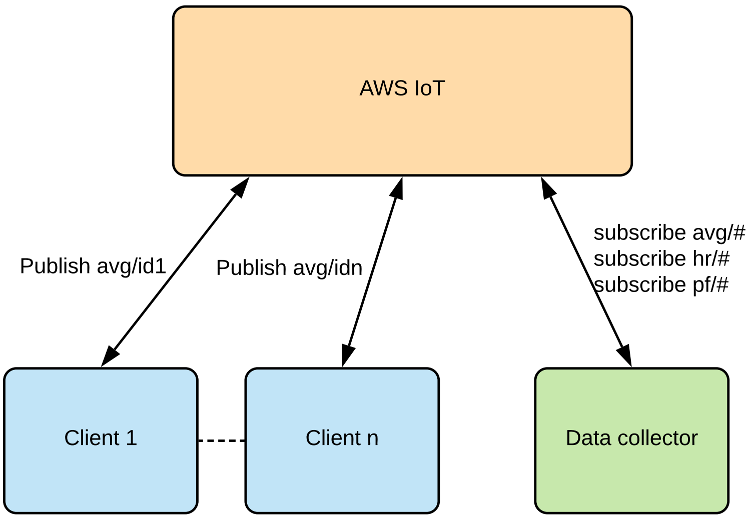


Figure 5.14: Illustration of the AMS data collection.

Chapter 6

Testing and Results

The main focus of this chapter is the testing and results of the ATmega system, as this was the system used for data collection. The system was tested and verified against the acceptance criteria in section 4.3. Graphs and plots of data collected are included at the end of this chapter. Some of the results in section 6.2 are taken from the data collection in section 6.3 as no data were saved from the tests described in section 6.1.

6.1 Testing procedure

6.1.1 PCB and hardware test

Voltage levels were tested by using a multimeter to measure the voltage at multiple places on the board. For the testing of the M-Bus converter, the UART0 of the ATmega was used to read in bytes and printing them to a PC serial console using a UART to USB converter chip. The ADC circuit was not tested as it was not used in this thesis work.

6.1.2 Power failure test

The power failure functionality testing included a test for the backup power to test against AC9, and a test of the power failure interrupt functionality testing against AC10. Backup power was tested by connecting a power supply and charging the supercapacitor for several minutes before removing power. Subsequently, the time before the system stopped functioning was observed. This was done by continuously printing text to a PC serial console and checking when the PC would stop to receive proper values. Criteria AC10 was tested by removing the cable to the AMS meter to simulate a power fail, and wait for the system to upload the voltage measurement data to the cloud, provided Wi-Fi was available.

6.1.3 Full System test

A full system test was performed using the smart meter provided by the supervisor and using a Wi-Fi network shared from a mobile phone. Certificates and keys for the TLS connection were downloaded from AWS IoT and flashed to the ESP8266 Wi-Fi module. In addition to monitoring incoming data from the AWS IoT management console, the data collector software was set up to run on a virtual private server (VPS) hosted at *vultr.com*.

The test included changing the sending frequency by publishing messages to the topic *freq* from the AWS IoT console with different values in the range 1-60 to test against AC6. Reconnect capabilities were tested by turning the Wi-Fi off and on and checking if the system would be able to reconnect to AWS IoT, to test against AC7 and AC8. Testing of the data collector software checked that the software correctly decoded the raw byte data received and stored the values in CSV files.

Connection to AWS IoT and subscribing and publishing messages with QoS1 tested the system against AC3 and AC4. Normal running conditions tested the system against AC2, AC5, and AC11. The criteria AC1 was tested when setting up systems for data collection in section 6.3.

6.1.4 ESP32 system

A full system test was not performed on this system. The hardware and AMS message reception were tested by connecting to an AMS meter and confirming the reception of the messages. Wi-Fi and MQTT functionality were tested by connecting the device to AWS and subscribing and publishing to different topics.

6.2 Testing results

Not all of the assembled PCBs ordered were tested, although, the hardware of all the tested boards have been confirmed working. The backup power test confirmed that the backup power lasted at least one second, which is longer than estimated in section 5.1.2. Correct uploading of power failure data to the cloud was confirmed by checking that the data collector received the data and stored it into a CSV file. An example of this is shown in table 6.1, which shows the power fail data containing the ten latest voltage values, received from a single-phase meter. The first value is the timestamp in UNIX time, and the second is the voltage value for the single-phase meter for that timestamp. A three-phase meter would have three voltage values instead of one.

The system was able to receive MQTT messages on the *freq* topic and subsequently change the sending frequency for the measurement values. Connecting to AWS IoT using the TLS protocol through the ESP8266 chip was successful. After a Wi-Fi disconnect, the system was able to reconnect to the Wi-Fi AP and AWS IoT.

```
1555567890,227
1555567900,227
1555567910,227
1555567920,227
1555567930,227
1555567940,228
1555567950,227
1555567960,227
1555567970,227
1555567980,227
```

Table 6.1: Power fail data for a single-phase Kamstrup meter.

AMS data were successfully sent to the cloud, and subsequently, received by the data logger software and stored in CSV files. Examples of this are shown in table 6.2 and table 6.3, which shows the average, minimum, and maximum values for both a three-phase and single-phase meter. The values are sent once every minute, which can be seen from the difference in the timestamp values. The three-phase meter data contains more values as there are three current values and three voltage values as opposed to a single-phase meter, which has only one of each. The values shown in table 6.2 and table 6.3 corresponds to the measurement values listed in Appendix A where active power in import direction is the first value after the timestamp. Hourly energy data received is shown in table 6.4.

```
1557012050,188,187,189,0,0,0,0,0,0,50,49,50,9,9,9,10,10,10,79,79,79,245,244,245,245,245,245,245,245,245
1557012110,187,186,188,0,0,0,0,0,0,49,49,50,9,9,9,10,10,10,79,79,79,244,244,245,245,244,245,245,245,245
1557012170,157,134,187,0,0,0,0,0,0,54,50,56,9,9,9,10,10,10,67,58,79,245,244,245,245,244,245,245,245,245
```

Table 6.2: Average, minumum and maximum values received from a three-phase Kamstrup meter.

```
1555975290,901,639,1782,0,0,0,166,165,166,0,0,0,395,284,772,233,232,234
1555975350,708,636,1058,0,0,0,165,164,166,0,0,0,313,283,460,233,233,234
1555975410,948,801,1242,0,0,0,213,157,233,0,0,0,424,367,548,233,233,233
```

Table 6.3: Average, minumum and maximum values received from a single-phase Kamstrup meter.

6.2.1 ESP32 system

The hardware was confirmed working together with Wi-Fi and MQTT functionality. The system could successfully subscribe, publish, and receive MQTT messages using the TLS protocol. The implementation of the FreeRTOS task for reading in AMS messages, converting the message to JSON and sending it to the cloud was only partially completed. The task implemented caused random crashes, and time limitations were the main reason for not completing the software. However, the individual components were confirmed working.

```
1555830005,3186295,0,534246,1076
1555833605,3186486,0,534302,1076
1555837205,3186641,0,534351,1076
```

Table 6.4: Hourly cumulative energy values received from a single-phase Kamstrup meter.

6.3 Data collection

6.3.1 Data sources

Three systems were used to collect AMS data from consumers described in table 6.5. Data were collected from mid-April to approximately the end of May. While two of the consumers had a Wi-Fi AP very close to the AMS meter, the Wi-Fi AP of the last house was not placed close to the AMS meter. Therefore, the system would occasionally get disconnected from the Wi-Fi AP. The meter numbers in table 6.5 are referred to throughout the rest of the thesis.

The consumers were not in close geographical proximity to each other, where the minimum air distance between any of them was approximately three kilometers. All the property owners have agreed to the usage of their AMS meter data in this thesis.

Number	Meter type	Description	Wi-Fi proximity
1	Three-phase Kamstrup	An apartment in a new apartment complex with one resident.	Very close
2	Single-phase Kamstrup	An older house with two residents.	Far
3	Three-phase Kamstrup	An older house with two residents.	Very close

Table 6.5: Description of the three consumers used for data collection.

Four additional systems were set up to collect data from *Granåsen Toppidrettssenter*. These systems were set up very late during the thesis work in collaboration with another master's student. Hence, data from these meters are not presented here. All these meters were three-phase meters from Kaifa.

A single system was also set up to acquire data from a smart meter at *Pirbadet* in Trondheim in collaboration with another master's student. The meter is located deep in the basement without the possibility of Wi-Fi or mobile signals. As a result, the system was connected directly to a PC through a UART to USB converter chip in order to store the measurement data locally on the PC.

6.3.2 Graphs

Most of the graphs in this section have been smoothed using a moving average filter to reduce noise and spikes in the graphs. The span of this moving average filter is set to the last 50 values for all graphs except fig. 6.7 which uses a span of 210 values. The graphs showing the correlation between variables in fig. 6.5 and fig. 6.8 use non-smoothed data. Correlation data is presented together with the Pearson Correlation Coefficient, which is a measure of the linear correlation between two variables. A correlation coefficient of 0 indicates no correlation, -1 indicates a total negative linear correlation, while 1 indicates a total positive linear correlation.

Figure 6.1 shows active power data collected from meter number 2 during April 28 and fig. 6.2 shows normalized average active power, current and voltage data from the same day. The values have been normalized between 0 and 1, where 0 is the minimum value from the data set, and 1 is the maximum value.

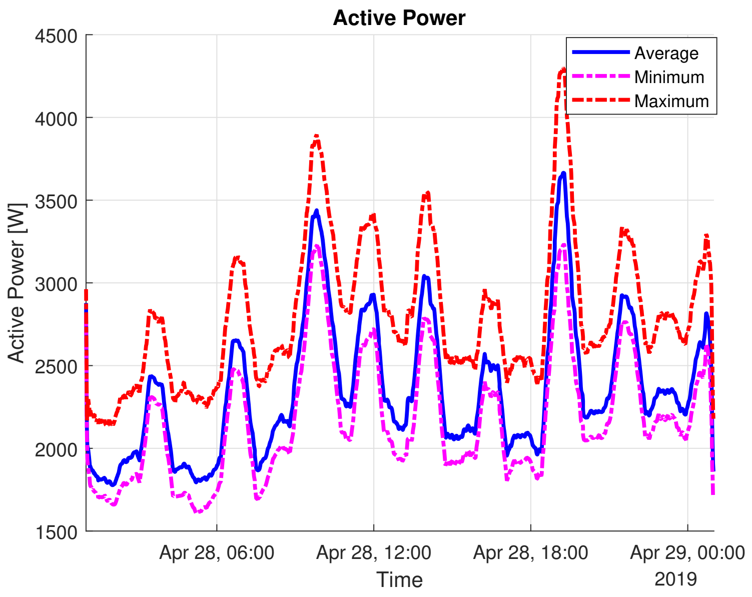


Figure 6.1: Average, minimum and maximum imported active power from meter 2 for one day.

In fig. 6.3 the average voltage values for all three meters is shown for April 28. Since the two three-phase meters contain one voltage value for each phase, the average of these three average values is calculated and used in the graph. Figure 6.4 shows the same data when each data set is normalized in the range 0 to 1 where 0 is minimum, and 1 is maximum. The plot in fig. 6.5 shows the correlation between the normalized voltage values shown in fig. 6.4 where V1 is the voltage values from meter 1, V2 is from meter 2 and V3 is from meter 3. Histograms of the variables are along the diagonal of the figure and scatter plots between the variables appear off the diagonal. The reason for the larger intervals

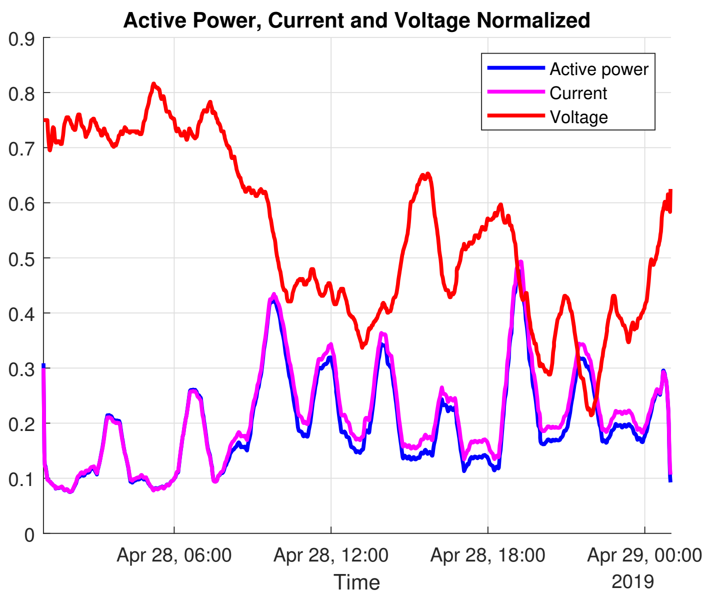


Figure 6.2: Normalized average imported active power, instantaneous current (RMS) and voltage from meter 2 for one day.

between the histogram bars for the data from meter 2, is that these voltage values are integer numbers while the values from meter 1 and meter 3 are not integer values because of the average calculation performed on these values.

The graphs in fig. 6.6 and fig. 6.7 shows average voltage and normalized average voltage data from meter 1 and meter 3 for one week. Correlation between the two data sets is shown in fig. 6.8 which uses the average voltage data shown in fig. 6.7. The histogram x-axis in fig. 6.8 and fig. 6.5 is somewhat misleading. The rightmost bar is always a bucket up to and including 1 while the leftmost is always a bucket from 0. This relation is not shown clearly in these two figures.

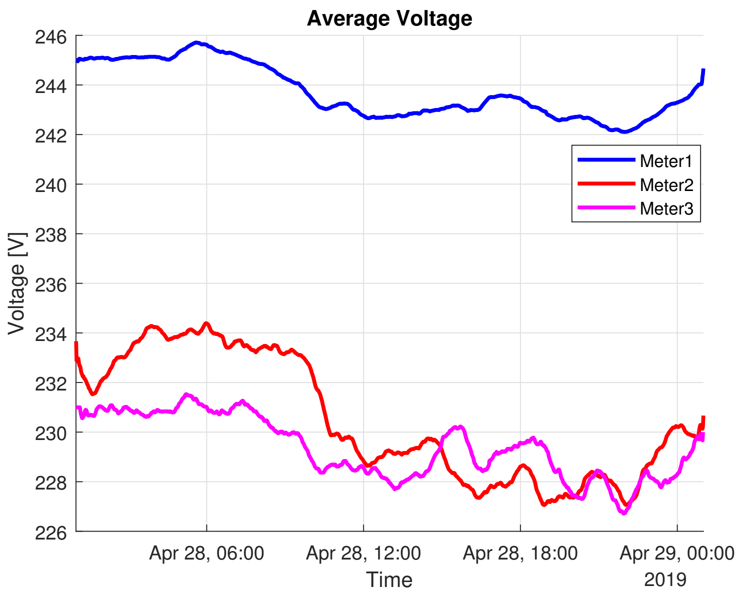


Figure 6.3: Average voltage from all three meters for one day.

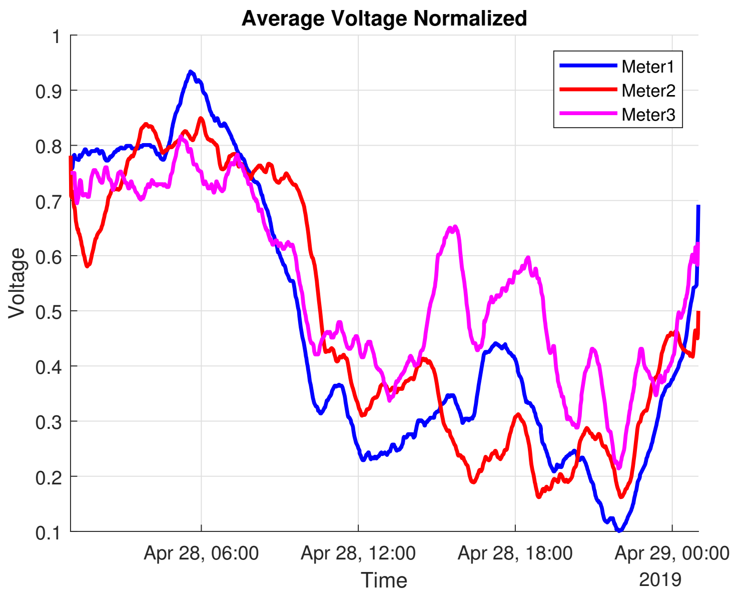


Figure 6.4: Normalized average voltage from all three meters for one day.

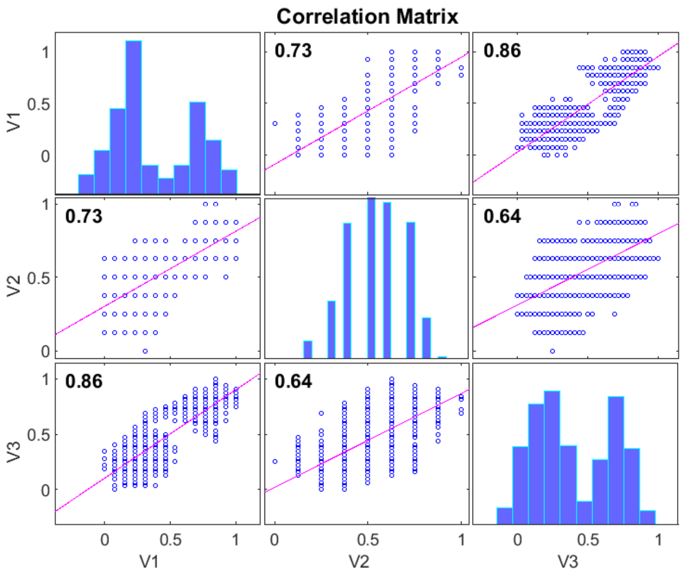


Figure 6.5: Correlation plot for the average voltage values in fig. 6.4.

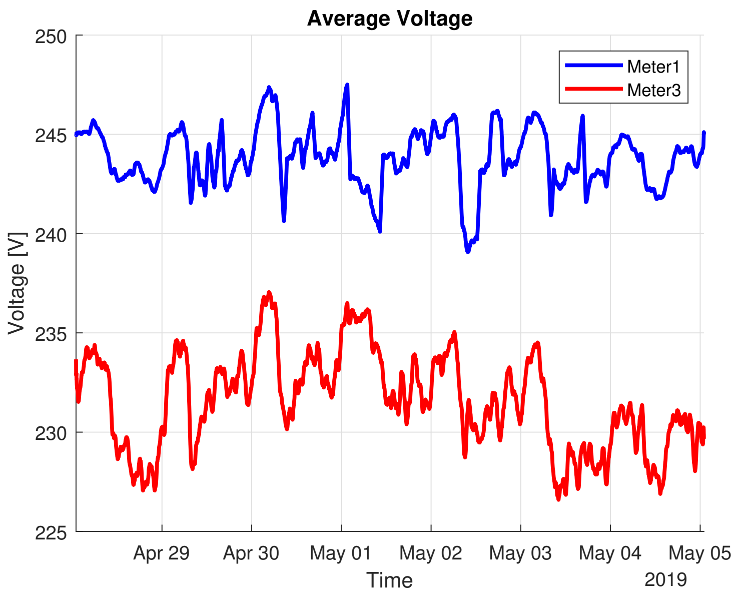


Figure 6.6: Average voltage from two meters for one week.

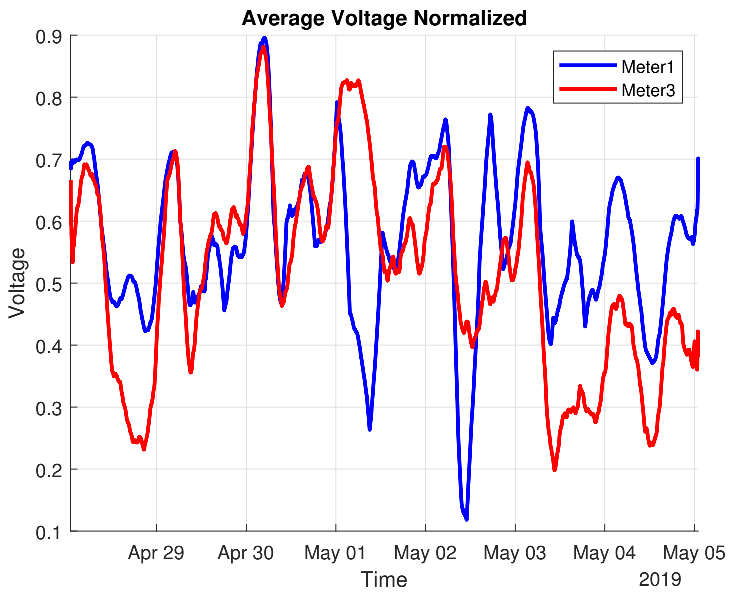


Figure 6.7: Normalized average voltage from two meters for one week.

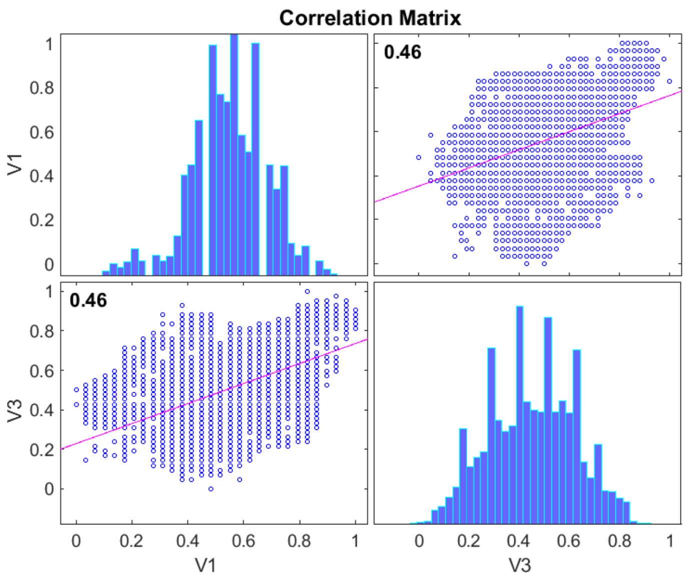


Figure 6.8: Correlation plot for the average voltage values in fig. 6.7.

Chapter 7

Discussion

The discussion chapter contains a discussion of the ATmega system results and performance, comparing it to the acceptance criteria set in section 4.3. A discussion of the results and potential of the ESP32 system is included. A discussion and a short analysis concerning the data collected and presented in section 6.3 is included. Furthermore, inconsistencies with the AMS standards are discussed, and lastly, a discussion on potential improvements is presented.

7.1 System results

The data collection and the testing results proved that the system was able to satisfy most of the acceptance criteria in section 4.3. AMS data were collected from single-phase and three-phase meters from both Kaifa and Kamstrup, proving that the system was able to work with different meter types, thus satisfying AC1. The data processing and AMS message reception were implemented successfully, which is shown from the data presented in fig. 6.1, satisfying AC2 and AC5. Both AC3 and AC4 were satisfied by connecting to AWS and subscribing and publishing messages using QoS1. The power fail functionality was proven working by the results of the testing, satisfying AC9 and AC10. However, one issue discussed in section 7.1.1 was discovered during data collection.

The ability to adjust the data sending frequency was successful during testing, thus satisfying AC6. During the data collection, the sending frequency was kept at the default value, which is set to once every minute. Criteria AC7 and AC8 were satisfied by the performance of the system connected to meter 2. Although frequently losing Wi-Fi connection, it was always able to reconnect and continue to send data. Network loss is discussed more thoroughly in section 7.1.2, where a connection issue is also discussed.

Criteria AC12 was satisfied by choosing the ATmega324PB as the MCU of the system.

The hourly energy values received from the AMS meters were successfully sent to the cloud, thus satisfying AC11.

7.1.1 Power failure

No power fail data were received from meter 1 and meter 3, while power failure data were received from meter 2 eleven times during the data collection. It is known for a fact that the household did not experience power outages as often as indicated by the power failure data, which means at least some of the power failures were false. Nothing of interest was found when inspecting the voltage values in the power failure data, which contained voltage values from the 90 seconds before a power failure.

It is unknown why the system reported this many power failures. The system would enter power failure mode only if the power failure interrupt pin on the TSS721A chip was activated, which limits the possible causes of these power fails. One possible explanation could be as simple as a weak cable from the AMS meter to the embedded system. This could possibly cause the M-Bus voltages to drop momentarily, causing the power failure functionality of the TSS721A to activate. Alternatively, it could be an issue with this specific meter or embedded hardware since the systems at the other two meters never sent in power failure data.

7.1.2 Loss of Wi-Fi connection

The system set up at meter 2 had a weak Wi-Fi connection causing the connection to drop out occasionally. Usually, it managed to reconnect quickly, although, on some occasions, the system lost connection for several hours. Despite the frequent loss of Wi-Fi connection, the system was always able to reconnect and never had to be manually reset, thus confirming that AC8 was satisfied.

Although two of the systems set up never had any problems with reconnecting, the system set up at meter 3 had to be manually restarted once. It was confirmed that the yellow LED was turned off, indicating that the system was in the *Not connected* state and unable to reconnect. Although the cause of this problem was not identified, one possible explanation is that the Wi-Fi module failed to reconnect to the Wi-Fi AP. Another possible cause could be an issue with the wireless router. After discovering this issue, a reconnect counter was added in the software. If this counter reaches a specific value, the Wi-Fi module will be reset by the ATmega.

7.1.3 ESP32 system

Although this system was not finished and will not be able to satisfy all of the acceptance criteria, the integrated Wi-Fi functionality results in several advantages compared to the system with the ATmega. The SDK available for the ESP32 chip allows to easily integrate several protocols and components into the software. The SDK support for the

MQTT protocol and Wi-Fi functionality made it easy to use these features for this system. Additionally, the SDK is very actively developed with new features and improvements getting added regularly. The SDK supports registering event handlers to handle specific events like a Wi-Fi event handler and MQTT event handler. In this way, automatic Wi-Fi reconnecting can easily be implemented. The MQTT event handler can be used to automatically subscribe to specific topics when the client manages to connect to the server. These features can result in the system satisfying AC8.

As a result of the usage of FreeRTOS in the SDK, multiple tasks can be created. The AMS message reception was separated into a task, which can satisfy AC7. The multi-core architecture also allows for running tasks in parallel in addition to concurrently. The large size of the flash memory and SRAM available makes it possible to implement more advanced functionality requiring more resources than what is possible with the ATmega324PB MCU. This was utilized by using the JSON format to send data to the cloud. By sending every single AMS message to the cloud, the data processing part is moved from the embedded system to the cloud solution.

This system does not include backup power to handle a power failure. Hence AC9 and AC10 are not satisfied. Criteria AC12 is satisfied by this system, except for differential ADC capabilities. The rest of the acceptance criteria could be considered satisfied if the software were to be finished, although, more development and testing of the system should have been conducted to verify this.

Despite not being used for data collection, this system contains a powerful processor and a large SDK which can be used to implement more complex software. An additional supported feature is over the air (OTA) updates, which means that firmware updates can be distributed to active systems through the internet. This feature can be very useful if a large number of active systems needs a software update. When collecting AMS data using Wi-Fi, this system might be considered a better solution than the ATmega system despite satisfying less of the acceptance criteria. If this system is to be used, it should be relatively easy to complete the embedded software and get a fully working system.

7.2 Data analysis

The main focus of the data presented in section 6.3 is voltage values, as these can be used to analyze the state and load of the distribution network. Other measurement values like instantaneous current, power values, and cumulative energy values are more of interest when analyzing a consumers power usage as opposed to analyzing the distribution network. Power values would be more interesting if data were collected from geographically adjacent consumers, as the relationship between power usage and voltage values could be analyzed for that area. The voltage values for an entire week was only plotted for meter 1 and meter 3 since the data sets contained considerably more data points than the data set from meter 2. The weak Wi-Fi connection from the system at meter 2 was the reason for this.

The graph in fig. 6.2 shows that the active power is very closely related to the instantaneous current draw as expected. Figure 6.3 and fig. 6.4 shows that the variation in the voltage values are very similar for all three meters. The correlation plot in fig. 6.5 shows that there is a high correlation between the voltage values from the different meters, especially between meter 1 and meter 3. All three correlation coefficients are significantly higher than 0, which confirms the strong correlation seen in the graphs.

A larger amount of data is shown in fig. 6.6 and fig. 6.7 which shows the average voltage values from meter 1 and meter 2 for one week. As can be seen from the graphs, there is a clear trend of higher voltage values at night and lower voltage levels in the morning and afternoon/evening. This trend is as expected since the distribution network load normally peaks during the times where the voltage is the lowest. The correlation in fig. 6.8 is not as high as fig. 6.5, however, the correlation coefficient is still significant enough to conclude that there is a clear trend between the two voltage values. If the correlation coefficient is computed using the smoothed data used to plot fig. 6.7, it increases from 0.46 to 0.54.

From the analysis of the collected data, it is apparent that there is a trend in the variation of the voltage values for the three meters. It is reasonable to assume that this trend is the same independent of geographical location, as it is highly related to the network load caused by power consumption. A higher network load causes a higher voltage drop from transformers to the consumers. The network load is lowest during the night when the least amount of power is consumed, and highest during the morning and afternoon/evening, which is when power consumption reaches a high point.

The methods for error detection and topology estimation presented in section 2.1 could not be applied to the data collected. A HIF as discussed in section 2.1 seldom happens, and the methods presented for detecting a HIF are best suited for simulations. Detecting topology errors in the distribution network require information concerning the topology data of the distribution operator, which is not freely available. However, the work done in [3] indicates that the smart meter data can be used for this purpose.

7.3 AMS standards inconsistencies

The inconsistencies between AMS suppliers in the implementation of the different AMS standards were also discussed in [14]. It seems as the three AMS meter suppliers have interpreted the DLMS/COSEM standards differently, which has resulted in different AMS message format for each supplier. This issue is confirmed by checking the different OBIS lists available at [2]. As an example of these inconsistencies, the OBIS codes for each measurement value is included in messages from Kamstrup and Aidon meters, while they are omitted from Kaifa messages. Furthermore, the Kamstrup and Aidon formats are not similar at all.

Another problem discussed in [14] was that the HDLC message format was not in accordance with the HDLC message format as described in IEC 62056-46. This issue was clarified in an email correspondence with "*Norsk Elektroteknisk Komite*" (NEK) which is responsible for the Norwegian AMS standard. Appendix D shows the email received from NEK with all names anonymized. The full version of the DLMS/COSEM standard is not freely available, and the chapter containing the information in Appendix D is not available in the free excerpt in [5].

Overall, these issues did not have a significant impact on the implementation. Making the software able to handle different meter types required some extra work. The initialization procedure will now handle detecting the meter type and initializing variables later used to extract meter values from the AMS messages. The implementation supports both Kaifa and Kamstrup meters, and support for Aidon meters can easily be added.

7.4 Improvement potential

Connectivity

The usage of Wi-Fi requires a Wi-Fi AP nearby to where the AMS meter is located. Another possibility is to use the cellular network instead as this can be considered more reliable than using Wi-Fi. The new *nRF91* series from *Nordic Semiconductor* is a system-in-package with integrated cellular network functionality which can be considered for this application. This would require the development of a new PCB containing a *nRF91* module as the processing unit. A disadvantage of 4G equipment is the increased price of components. Alternatively, if neither Wi-Fi or 4G reception is available, a form of local flash storage could be used to hold some measurement values. This option would also require a redesign of the PCB.

If Wi-Fi is to be used for data collection, the system containing the ESP32 chip can be considered more reliable and has more features and possibilities than the ATmega system used for data collection in this thesis. The completion of development and testing and possibly redesigning the ESP32 system should be considered if Wi-Fi is going to be used.

Despite the weaknesses of using Wi-Fi, it has some advantages as well. If a smart meter is located in a basement or another location without cellular reception, Wi-Fi might still be available. The systems set up at *Granåsen Toppidrettssenter* are located in a room in the basement with limited cellular reception. The systems were here connected to a 4G LTE modem using Wi-Fi. It is unknown if a built-in cellular modem and antenna on the PCB would be powerful enough to connect to the cellular network from this room. The 4G LTE modem used was able to establish a connection despite the weak cellular reception in the room.

Software ATmega

Although the ATmega system was able to satisfy all acceptance criteria, there is potential for improving the software. One rare bug was detected, which caused the system to send the same MQTT message twice to the cloud. As a result, the python script used to store measurement values to CSV files, stored two identical measurements. The source of this bug was not found. However, it did not cause any other problems. There are several modules and sections of software with improvement potential, where most of them have been marked with a "*TODO*" comment in the source code.

Hardware

The possibility of powering the system directly from the M-Bus lines should be explored. Drawing power solely from the M-Bus lines would most likely require a significant re-design where the TSS721A chip is dropped, as its maximum LDO regulator current output is less than 1mA. Furthermore, if the PCB should contain either cellular or Wi-Fi capabilities, drawing power from the M-Bus would require the usage of sleep or low-power modes for the processing unit and a chargeable power source on the board. This is required since the maximum current which can be drawn from the meter can be as low as 6mA for Kamstrup meters. However, the solution reviewed in section 2.2 shows that this is possible with a Wi-Fi module. Powering directly from the M-Bus is advantageous as there would be no need for an external power supply, and only one cable would be needed to set up the system. Lastly, a protective cover should be produced to protect the hardware from damage and coming into contact with other surfaces.

Utilization of Cloud service

The AWS cloud service was mainly used as an MQTT server and as a monitoring tool in this thesis work. The other possibilities of this cloud service can be explored. Alternatively, a different cloud service could be used. Data analysis and visualization in the cloud should be looked into, although the data collection scheme storing data in CSV files were very useful for data analysis. Real-time visualization of measurement data performed by the cloud service could be useful for further monitoring.

Chapter 8

Conclusion

Two embedded system designs for data acquisition from AMS meters have been suggested and implemented in this thesis. Both systems can receive and process data from an AMS meter and send this data to a cloud service using Wi-Fi and the MQTT protocol over a secure TLS connection.

Several complete assembled boards were ordered for the ATmega system, which were used for data collection from multiple sources. The software development for the alternative ESP32 system was not finished, while the hardware of this system was confirmed working. The data collection was implemented by calculating the average, minimum, and maximum values of all the meter measurement values within some time interval. Subsequently, these values were sent to the cloud solution and received by a program which stored the values in CSV files.

All the acceptance criteria defined for the complete system were mostly satisfied by the ATmega system used for data collection. However, some smaller issues were found related to loss of Wi-Fi connection and the power failure functionality of the system. Overall, these issues did not impact the results of data collection in a significant way.

Although not enough data were collected to perform a more extensive analysis of the distribution network, a significant trend in the variation in voltage values at consumers was found from the collected data. A larger scale data collection from consumers in close proximity to each other should be performed to be able to perform a more thorough analysis of the surrounding distribution network.

Chapter 9

Further work

The list of suggested work presented below is mostly based on the discussions in section 7.4. Two of the points are related to data collection in order to analyze the distribution network using the AMS data.

Suggestions

- A redesign of hardware containing a built-in cellular module for communicating with the cloud.
- Look into the possibility of powering the board from the M-Bus lines.
- Utilize features of a cloud service for visualization and analysis in real-time.
- Collect data from more consumers, preferably in close proximity and connected to the same transformer.
- Collect data from a consumer delivering power to the distribution network through local energy production.
- Create a cover enclosing the PCB to protect the hardware.

Bibliography

[1] Ala Al-Fuqaha, Mohsen Guizani, Mehdi Mohammadi, Mohammed Aledhari, and Moussa Ayyash. Internet of Things: A Survey on Enabling Technologies, Protocols, and Applications. *IEEE Communications Surveys & Tutorials*, 17(4):2347–2376, 2015.

[2] Informasjon til utviklere — Norsk Elektroteknisk Komite (NEK). `https://www.nek.no/info-ams-han-utviklere/`. Last visited 2019-05-20.

[3] Anders Hylen Klippenberg. Bruk av nye sensorer og AMS i distribusjonsnettet for a validere netttopologi. Master's thesis, NTNU, 2018.

[4] DLMS User Association. COSEM Interface Classes and OBIS Object Identification System/Blue Book (Excerpt). `https://www.dlms.com/files/Blue-Book-Ed-122-Excerpt.pdf`. Last visited 2019-03-18.

[5] DLMS User Association. DLMS/COSEM Architecture and Protocols/Green Book (Excerpt). `https://www.dlms.com/files/Green-Book-Ed-83-Excerpt.pdf`. Last visited 2019-03-18.

[6] Soham Chakraborty and Sarasij Das. Application of Smart Meters in High Impedance Fault Detection on Distribution Systems. *IEEE Transactions on Smart Grid*, 10(3):3465–3473, may 2019.

[7] Espressif Systems. ESP32 Series Datasheet. `https://www.espressif.com/sites/default/files/documentation/esp32_datasheet_en.pdf`. Last visited 2019-05-10.

[8] H. Farhangi. The path of the smart grid. *IEEE Power and Energy Magazine*, 8(1):18–28, jan 2010.

[9] Amin Ghaderi, Herbert L. Ginn, and Hossein Ali Mohammadpour. High impedance fault detection: A review. *Electric Power Systems Research*, 143:376–388, feb 2017.

[10] Amin Ghaderi, Hossein Ali Mohammadpour, Herbert L. Ginn, and Yong-June

Shin. High-Impedance Fault Detection in the Distribution Network Using the Time-Frequency-Based Algorithm. *IEEE Transactions on Power Delivery*, 30(3):1260–1268, jun 2015.

[11] Andy Gock. avr-uart. `https://github.com/andygock/avr-uart`. Last visited 2019-05-24.

[12] Network Working Group. PPP in HDLC-like Framing. `https://tools.ietf.org/html/rfc1662`. Last visited 2019-04-25.

[13] Internet Engineering Task Force (IETF). The Transport Layer Security (TLS) Protocol Version 1.3. `https://tools.ietf.org/html/rfc8446`. Last visited 2019-04-25.

[14] Marius Lervik. Development of an embedded system for reading a wired M-Bus. Specialization project report, 2018.

[15] Microchip. ATmega324PB Datasheet. `http://ww1.microchip.com/downloads/en/DeviceDoc/40001908A.pdf`. Last visited 2019-05-07.

[16] Energi Norge. Nettstruktur og organisering. `https://www.energinorge.no/fagomrader/stromnett/kraftsystemet/nettstruktur-og-organisering/`. Last visited 2019-03-07.

[17] NVE. Smarte strømmålere (AMS). `https://www.nve.no/stromkunde/smarte-strommalere-ams/`. Last visited 2019-03-21.

[18] OASIS. MQTT documentation. `http://docs.oasis-open.org/mqtt/mqtt/`. Last visited 2019-04-25.

[19] Øivind Kristian Rue. Smarte Nett. `https://www.statnett.no/contentassets/18e85c65282144d4af72f7ccf805cf2c/smarte-nett.pdf`. Last visited 2019-03-14.

[20] Electricity grid simple. `https://commons.wikimedia.org/wiki/File:Electricity_grid_simple-_North_America.svg`. Last visited 2019-05-09.

[21] Tibber Pulse - MQTT? - Strømsparing - Hjemmeautomasjon. `https://www.hjemmeautomasjon.no/forums/topic/4255-tibber-pulse-mqtt/`. Last visited 2019-03-17.

[22] M-Bus Usergroup. The M-Bus: A Documentation Rev. 4.8. `http://www.m-bus.com/mbusdoc/default.php`. Last visited 2019-01-25.

[23] Francinei L. Vieira, Jose M. C. Filho, Paulo M. Silveira, Carlos A. V. Guerrero, and Marino P. Leite. High impedance fault detection and location in distribution networks using smart meters. In *2018 18th International Conference on Harmonics and Quality of Power (ICHQP)*, pages 1–6. IEEE, may 2018.

[24] C.G. Wester. High impedance fault detection on distribution systems. In *1998 Rural*

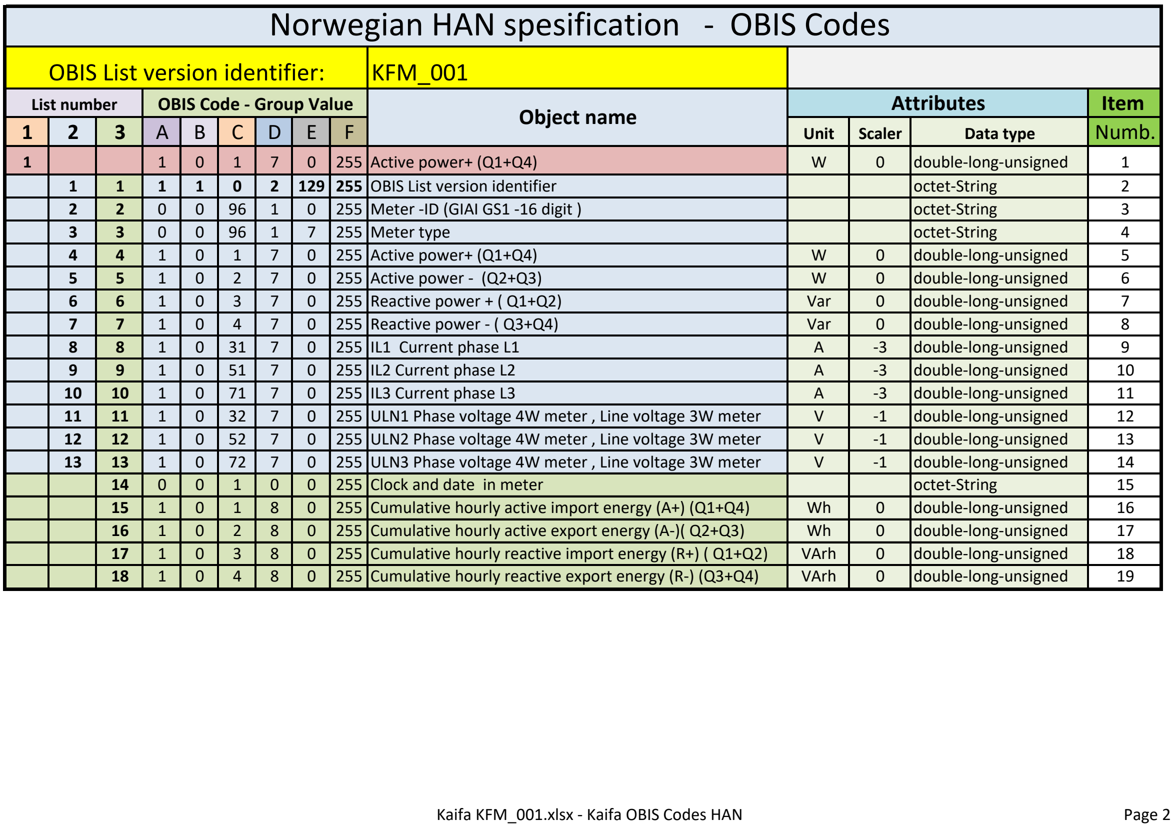
Electric Power Conference Presented at 42nd Annual Conference, pages c5–1–5. IEEE, 1998.

[25] Tetsuya Yokotani and Yuya Sasaki. Comparison with HTTP and MQTT on required network resources for IoT. In *2016 International Conference on Control, Electronics, Renewable Energy and Communications (ICCEREC)*, pages 1–6. IEEE, sep 2016.

Appendix A: Kaifa HAN specification

Norwegian HAN spesification - OBIS List Information

Item	Description	Value	Remarks
A	File name	KFM_001.xlsx	Filename : OBIS List identifier.xlsx . Format for publication is pdf.
B	List version - date	09.11.2018	DD.MM.YYYY
C	OBIS List version identifier	**KFM_001**	Shall be identical to corresponding OBIS code value in the meter
D	Meter type	All	
J	Baudrate M-BUS (HAN)	2400	
K	List 1 Stream out every	2 seconds	
M	List 2 Stream out every	10 seconds	
N	List 3 Stream out every	1 hour	The values is generated at XX:00:00 and streamed out from the HAN interface 10 seconds later (XX:00:10)
O	HAN maximum power to HEMS (mW)	500 mW	The largest power that the customer equipment (HEMS or display) can consume from the meter HAN interface
P	HAN maximum current to HEMS (mA)	21 mA	



Norwegian HAN spesification - OBIS Codes														
OBIS List version identifier:									KFM_001					
List number			OBIS Code - Group Value						Object name	Attributes			Item
1	2	3	A	B	C	D	E	F		Unit	Scaler	Data type	Numb.
1			1	0	1	7	0	255	Active power+ (Q1+Q4)	W	0	double-long-unsigned	1
	1	**1**	**1**	**1**	**0**	**2**	**129**	**255**	OBIS List version identifier			octet-String	2
	2	**2**	0	0	96	1	0	255	Meter -ID (GIAI GS1 -16 digit)			octet-String	3
	3	**3**	0	0	96	1	7	255	Meter type			octet-String	4
	4	**4**	1	0	1	7	0	255	Active power+ (Q1+Q4)	W	0	double-long-unsigned	5
	5	**5**	1	0	2	7	0	255	Active power - (Q2+Q3)	W	0	double-long-unsigned	6
	6	**6**	1	0	3	7	0	255	Reactive power + (Q1+Q2)	Var	0	double-long-unsigned	7
	7	**7**	1	0	4	7	0	255	Reactive power - (Q3+Q4)	Var	0	double-long-unsigned	8
	8	**8**	1	0	31	7	0	255	IL1 Current phase L1	A	-3	double-long-unsigned	9
	9	**9**	1	0	51	7	0	255	IL2 Current phase L2	A	-3	double-long-unsigned	10
	10	**10**	1	0	71	7	0	255	IL3 Current phase L3	A	-3	double-long-unsigned	11
	11	**11**	1	0	32	7	0	255	ULN1 Phase voltage 4W meter , Line voltage 3W meter	V	-1	double-long-unsigned	12
	12	**12**	1	0	52	7	0	255	ULN2 Phase voltage 4W meter , Line voltage 3W meter	V	-1	double-long-unsigned	13
	13	**13**	1	0	72	7	0	255	ULN3 Phase voltage 4W meter , Line voltage 3W meter	V	-1	double-long-unsigned	14
		14	0	0	1	0	0	255	Clock and date in meter			octet-String	15
		15	1	0	1	8	0	255	Cumulative hourly active import energy (A+) (Q1+Q4)	Wh	0	double-long-unsigned	16
		16	1	0	2	8	0	255	Cumulative hourly active export energy (A-)(Q2+Q3)	Wh	0	double-long-unsigned	17
		17	1	0	3	8	0	255	Cumulative hourly reactive import energy (R+) (Q1+Q2)	VArh	0	double-long-unsigned	18
		18	1	0	4	8	0	255	Cumulative hourly reactive export energy (R-) (Q3+Q4)	VArh	0	double-long-unsigned	19

Kaifa KFM_001.xlsx - Kaifa OBIS Codes HAN Page 2

OBIS codes available in different meter types										Meter Types				
OBIS List version identifier:									KFM_001					
List number			OBIS Code - Group Value						Object name	MA105H2E	MA304H3E	MA304H4	MA304T3	MA304T4
1	2	3	A	B	C	D	E	F						
1			1	0	1	7	0	255	Active power+ (Q1+Q4)					
	1	1	1	1	0	2	129	255	OBIS List version identifier	X	X	X	X	X
	2	2	0	0	96	1	0	255	Meter -ID (GIAI GS1 -16 digit)	X	X	X	X	X
	3	3	0	0	96	1	7	255	Meter type	X	X	X	X	X
	4	4	1	0	1	7	0	255	Active power+ (Q1+Q4)	X	X	X	X	X
	5	5	1	0	2	7	0	255	Active power - (Q2+Q3)	X	X	X	X	X
	6	6	1	0	3	7	0	255	Reactive power + (Q1+Q2)	X	X	X	X	X
	7	7	1	0	4	7	0	255	Reactive power - (Q3+Q4)	X	X	X	X	X
	8	8	1	0	31	7	0	255	IL1 Current phase L1	X	X	X	X	X
	9	9	1	0	51	7	0	255	IL2 Current phase L2	NA	X	X	M1	X
	10	10	1	0	71	7	0	255	IL3 Current phase L3	NA	X	X	X	X
	11	11	1	0	32	7	0	255	ULN1 Phase voltage 4W meter , Line voltage 3W meter	X	X	X	X	X
	12	12	1	0	52	7	0	255	ULN2 Phase voltage 4W meter , Line voltage 3W meter	NA	M2	X	M2	X
	13	13	1	0	72	7	0	255	ULN3 Phase voltage 4W meter , Line voltage 3W meter	NA	X	X	X	X
		14	0	0	1	0	0	255	Clock and date in meter	X	X	X	X	X
		15	1	0	1	8	0	255	Cumulative hourly active import energy (A+) (Q1+Q4)	X	X	X	X	X
		16	1	0	2	8	0	255	Cumulative hourly active export energy (A-)(Q2+Q3)	X	X	X	X	X
		17	1	0	3	8	0	255	Cumulative hourly reactive import energy (R+) (Q1+Q2)	X	X	X	X	X
		18	1	0	4	8	0	255	Cumulative hourly reactive export energy (R-) (Q3+Q4)	X	X	X	X	X

Meter types			
Meter type	Voltage	Current	Connection
MA105H2E	1x230 V	5(80) A	DCM
MA304H3E	3x230 V	5(100)A	DCM
MA304H4	3x230/400V	5(100)A	DCM
MA304T3	3x230 V	1(6)A	CTM
MA304T4	3x230/400V	1(6)A	CTM

M1	Value is alllways 0 A
M2	Value is alllways 0 V

Kaifa KFM_001.xlsx - Kaifa OBIS Codes HAN Page 3

Norwegian HAN spesification - OBIS Codes	
Item	
Number	Long description OBIS Code
1	Active power in import direction (W)
2	Version number of this OBIS list to track the changes
3	Serial number of the meter point:16 digits (69706314xxxxxxxx)
4	Type number of the meter: For example "MA304H3E"
5	Active power in import direction (W)
6	Active power in export direction (W)
7	Reactive power in import direction (VAr)
8	Reactive power in export direction (VAr)
9	Instantaneous current of L1(mA) (RMS Value based on 1 second integration period)
10	Instantaneous current of L2 (mA) (RMS Value based on 1 second integration period)
11	Instantaneous current of L3 (mA) (RMS Value based on 1 second integration period)
12	Instantaneous voltage L1-L2 (Phase voltage 4W meter , Line voltage 3W meter) (dV / 0,1V) (RMS Value based on 1 second integration period)
13	Instantaneous voltage L1-L3 (Phase voltage 4W meter , Line voltage 3W meter) (dV / 0,1V) (RMS Value based on 1 second integration period)
14	Instantaneous voltage L2-L3 (Phase voltage 4W meter , Line voltage 3W meter) (dV / 0,1V) (RMS Value based on 1 second integration period)
15	Local date and time of Norway (Winter: CET (UTC+1) - Summer: CEST (UTC+2)) http://www.timeanddate.com/worldclock/norway/oslo
16	Cumulativeactive import active energy (A+) displayed hourly
17	Cumulativeactive export active energy (A-) displayed hourly
18	Cumulativeactive import reactive energy (R+) displayed hourly
19	Cumulativeactive export reactive energy (R-) displayed hourly

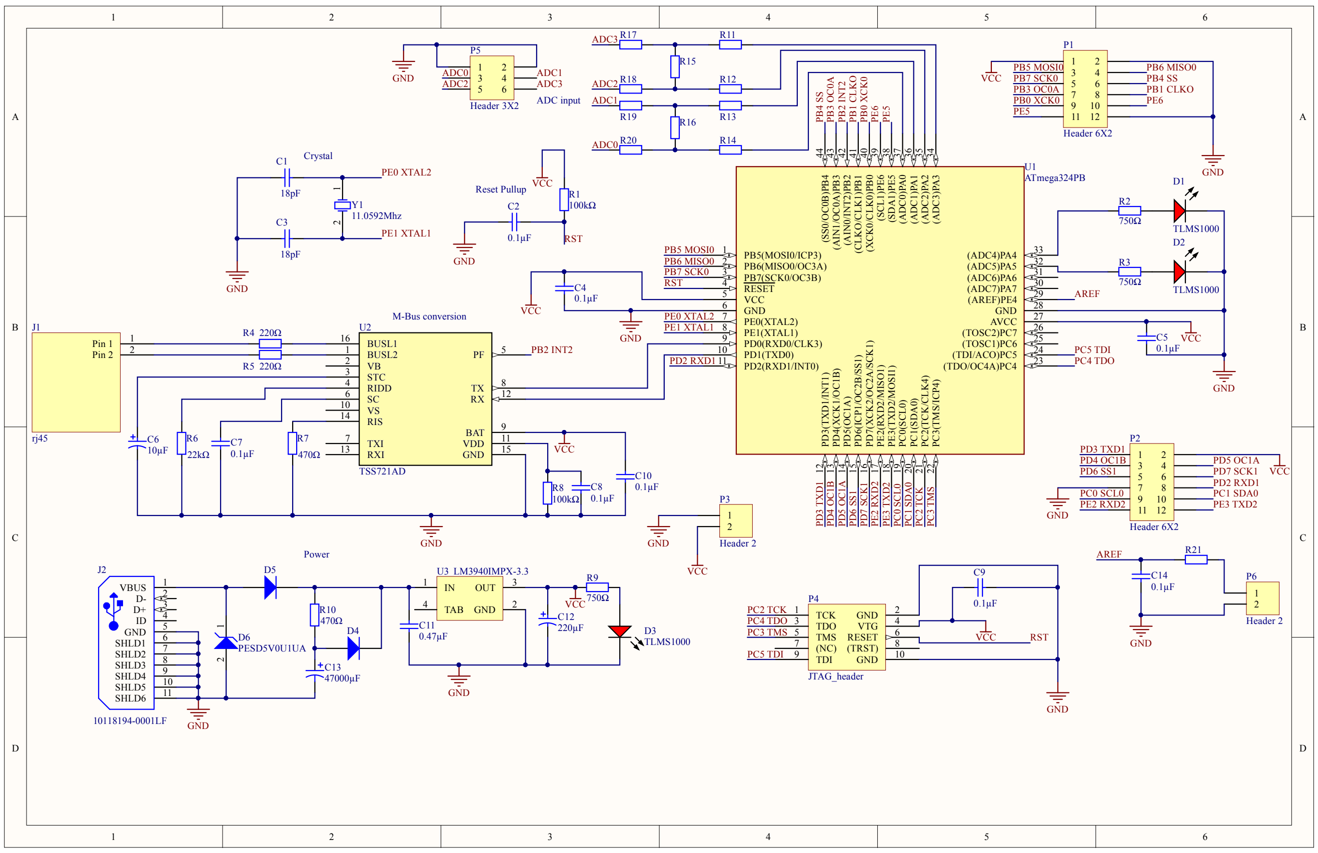
Kaifa KFM_001.xlsx - Kaifa OBIS Codes HAN Page 4

List Interval

Clock	List interval		
	2 sec	10 sec	3600 sec
14:59:56	List 1		
14:59:58	List 1		
15:00:00		List 2	
15:00:02	List 1		
15:00:04	List 1		
15:00:06	List 1		
15:00:08	List 1		
15:00:10			List 3
15:00:12	List 1		
15:00:14	List 1		
15:00:16	List 1		
15:00:18	List 1		
15:00:20		List 2	
15:00:22	List 1		

Appendix B: Hardware ATmega

B1: Schematic

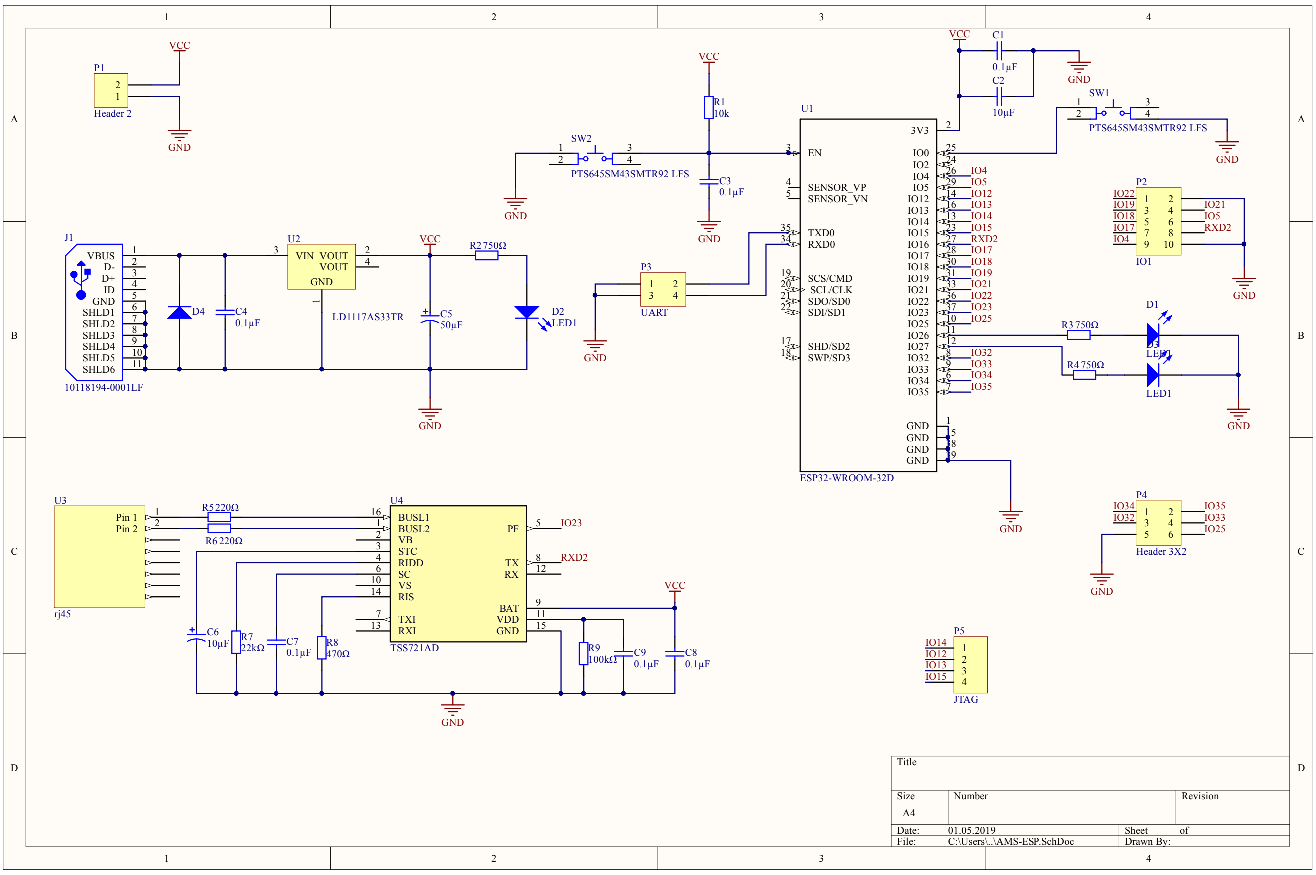


B2: Component list

Designator	Value	Description	Package
C1,C3	18pF	Capacitor	0603
C2,C4,C5,C7-C10,C14	0.1uF	Capacitor	0603
C6	10uF	Capacitor	Radial Can SMD, D=4mm
C11	0.47uF	Capacitor	0603
C12	220uF	Capacitor	Radial Can SMD, D=6.3mm
C13	47000uF	SuperCapacitor	Radial, Can TH
D1	2mA	Red LED	0603
D2	2mA	Yellow LED	0603
D3	2mA	Orange LED	0603
D4,D5	15V,1A	LSM115JE3/TR13 Schottky Diode	DO-214BA
D6		TVS Diode	SOD-323
J1		Rj45 Conn	
J2		Micro USB Conn	
P1,P2	HDR,6x2	Headers	Female, 2.54mm
P3	HDR,1x2	Headers	Female, 2.54mm
P4	HDR,5x2	Headers	Male, 1.27mm
P5	HDR,3x2	Headers	2.54mm
P6	HDR,1x2	Headers	2.54mm
R1,R8	100kΩ	Resistor	0603
R2,R3,R9	750Ω	Resistor	0603
R4,R5	220Ω, 1/2W	Resistor	0805
R6	22kΩ	Resistor	0603
R7,R10	470Ω	Resistor	0603
U1	ATMega324PB	MCU	44-TQFP
U2	TSS721AD	M-Bus slave	16-SOIC
U3	3.3V 1A	LM3940IMPX LDO regulator	SOT-223-4
Y1	11.0592Mhz	Crystal oscillator	HC-49/US SMD

Appendix C: Hardware ESP32

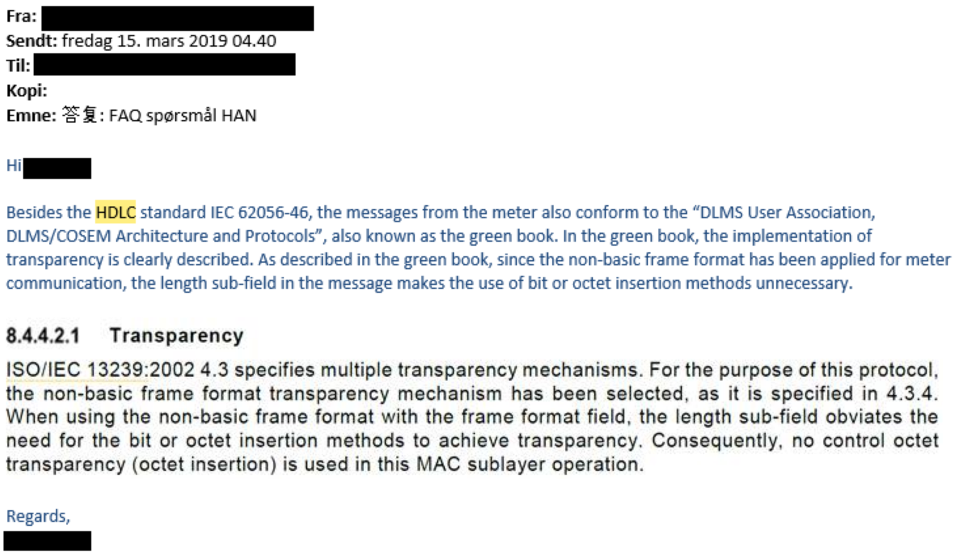
C1: Schematic



C2: Component list

Designator	Value	Description	Package
C1,C3,C4,C7-C9	0.1uF	Capacitor	0805
C2	10uF	Capacitor	0805
C5	50uF	Capacitor	Radial Can SMD, D=4mm
C6	10uF	Capacitor	Radial Can SMD, D=4mm
D1,D2,D3	2mA	LED	0805
D4		TVS diode	0805
J1		Rj45 Conn	
J2		Micro USB Conn	
P1	HDR,1x2	Headers	2.54mm
P2	HDR,5x2	Headers	2.54mm
P3	HDR,2x2	Headers	2.54mm
P4	HDR,3x2	Headers	2.54mm
P5	HDR,1x4	Headers	2.54mm
R1	10kΩ	Resistor	0805
R2,R3,R4	750Ω	Resistor	0805
R5,R6	220Ω, 1/2W	Resistor	0805
R7	22kΩ	Resistor	0805
R8	470Ω	Resistor	0805
R9	100kΩ	Resistor	0805
SW1,SW2		Tactile switch button	SMD
U1		ESP32-WROOM-32D	
U2	3.3V 1A	LD1117AS33TR LDO regulator	SOT-223-4
U3	TSS721AD	M-Bus slave	16-SOIC

Appendix D: NEK email



Fra: [redacted]
Sendt: fredag 15. mars 2019 04.40
Til: [redacted]
Kopi:
Emne: 答复: FAQ spørsmål HAN

Hi [redacted]

Besides the HDLC standard IEC 62056-46, the messages from the meter also conform to the "DLMS User Association, DLMS/COSEM Architecture and Protocols", also known as the green book. In the green book, the implementation of transparency is clearly described. As described in the green book, since the non-basic frame format has been applied for meter communication, the length sub-field in the message makes the use of bit or octet insertion methods unnecessary.

8.4.4.2.1 Transparency

ISO/IEC 13239:2002 4.3 specifies multiple transparency mechanisms. For the purpose of this protocol, the non-basic frame format transparency mechanism has been selected, as it is specified in 4.3.4. When using the non-basic frame format with the frame format field, the length sub-field obviates the need for the bit or octet insertion methods to achieve transparency. Consequently, no control octet transparency (octet insertion) is used in this MAC sublayer operation.

Regards,
[redacted]