NTNU
Norwegian University of
Science and Technology

Data gathering and -assembling from several smart meter HAN ports.

Erlend Grande

Master of Science in Cybernetics and Robotics
Submission date: June 2018
Supervisor: Geir Mathisen, ITK

Norwegian University of Science and Technology
Department of Engineering Cybernetics

NORWEGIAN UNIVERSITY OF SCIENCE AND TECHNOLOGY

DEPARTMENT OF ENGINEERING CYBERNETICS

Data gathering and -assembling from several smart meter HAN ports.

By:

Erlend Grande
erlengra@stud.ntnu.no

Supervisor: **Geir Mathisen**

NTNU

June 11, 2018

NTNU
Norges teknisk-naturvitenskapelige universitet

Fakultet for Informasjonsteknologi og Elektroteknikk
Institutt for Teknisk Kybernetikk

MASTER THESIS DESCRIPTION SHEET

Kandidat:	**Erlend Grande**
Emne:	**TTK4990**
Oppgavetittel (Norsk):	**Innsamling og sammenstilling av data fra HAN-porten til flere smarte strømmålere.**
Thesis title (English):	**Data gathering and -assembling from several smart meter HAN ports.**

Thesis description:

Normally, the real-time data from the smart meters are used for analyzing the in-house consumption and for in-house energy management systems. However, by analyzing real-time data from more smart meters in a neighborhood, information about the condition of the outdoor connecting grid can be extracted. As the smart meters already are installed in the grid due to fiscal reasons, these measurements are ”nearly free”. We want to develop an efficient system for collecting (near) real-time data from the HAN (home area network) port of distributed smart meters. The information shall be centralized for preliminary analysis, with focus on the condition of the connected grid.

The tasks will be:

- Conduct a litterary study to establish a theoretical basis within distributed data acquisition. Investigate also the possibilities measurements from more smart meters give for condition monitoring of the connected grid.

- Propose a system for near real-time acquisition of data from the HAN-port of smart meters. The proposed system shall, by use of data from more smart meters, give a centralized interpretation of the condition of the connected grid.
- As far as time permits, implement the suggested system.

Oppgave gitt: 8. Januar 2018
Oppgavens leveringsfrist: 11. Juni 2018

FAKULTET FOR INFORMASJONSTEKNOLOGI OG ELEKTRONTEKNIKK IE
INSTITUTT FOR TEKNISK KYBERNETIKK ITK

Abstract

Advanced electronic measuring devices, targeted at measuring power consumption, is replacing the older electro-mechanical metering devices. These devices, abbreviated AMS devices, are considered a significant infrastructure advancement from the conventional measuring of power, as the device is capable of communicating the readings directly to the distribution network manager for monitoring and billing in real time. Although automated power readout and data transfer to a centralized manager is a considerable improvement to the distribution network infrastructure, further advancement is necessary to allow distributed power generation to be incorporated into the radial low-voltage segment of the distribution network and to extend real-time monitoring abilities outside the current boundaries of the distribution network. These properties can enhance measurement quality and reduce measuring errors, allow for real-time regulation of transmission line loads and improve quality to the overall distribution network. A system was designed and implemented, utilizing the Meter-Bus protocol on the Home Area Network (HAN)-port on Kaifa Advanced Metering System (AMS) meters. The system module was designed to log and transmit AMS data read-out through the use of an integrated wireless transmitter and provide cloud-supported availability to the consumer and the distribution network manager. To achieve the notion of multiple devices communicating in real-time, an AMS simulator was necessary to provide data to the modules. This M-Bus simulator was successfully capable of acting as a functional AMS device and therefore allowing aggregated data availability from a decentralized Amazon Web Service (AWS) cloud client.

Sammendrag

Det pågår et teknologisk skifte i hvordan en foretar målinger av strømforbruket til den enkelte husstand. De gamle elektromekansike strømmålerne er i ferd med å fases ut til fordel for nye Avanserte Måle- og Styringsenheter. Disse nye målerne, kalt AMSer ansees å kunne forbedre infrastrukturen til distribusjonsnettet da målerne er i stand til å foreta automatiske avlesninger av effektforbruket til hver node i det radielle distribusjonsnettet og kommunisere denne dataen til nettleverandøren. Informasjonen kan benyttes til å produsere mer nøyaktige avgiftsmodeller eller generer et påløpende avgiftsmønster i sanntid. Selv om automatisert avlesning i seg selv regnes som en betydelig forbedring av distribusjonsnettet, så vil en ytterligere utnyttelse av denne teknologien kunne gi betydelige løft til den eldrende infrastrukturen til distribusjonsnettet. Å gjøre lavspenningsnettet transparent har lengen vært ønsket av nettopp nettleverandørene. Monitoreringsegenskaper utover trafostasjonene vil kunne bidra til å danne en last-profil av nettet, slik at fluktuasjoner i forbruk, fortetning på nettet og redusert strømkvalitet kan motvirkes ved lastregulering. En datalogger ble designet for å kunne motta og tolke informasjon fra en AMS-måler. Informasjon fra AMS-målerne er gjort tilgjengelig via den integrerte HAN-porten på enheten. Logge-enheten ble integrert med en trådløs kommunikasjonsløsning for å kunne laste opp mottatt informasjon til en skytjeneste. Fra skyen kan forbrukeren eller nettleverandøren monitorere forbruket i boligen AMS-enheten er installert som, noe vil kunne bidra til å gjøre distribusjonsnettet transparent. For å kunne se på hvordan flere AMSer med tilhørende loggerenheter kan tilby informasjon til sky-tjenesten ble det ytterligere konstruert en simulator, tiltenkt å kunne erstatte tilstedeværelsen av en funksjonell AMS-enhet under utviklingsfasen. AMS simulatorenen og logger-enheten var i stand til å lagre innhøstet informasjon i skyen og man dermed få et innblikk i hvordan sanntids måledata kan forbedre distribusjonsnettet vårt og tilnærme nette status som et ”smart nettverk”.

Preface

This project is written at the Department of Cybernetics(ITK), a subdivision at the Faculty of Information Technology and Electrical Engineering(IE) at the Norwegian University of Science and Technology(NTNU). The work is carried out with assistance and equipment from Sintef Department of Cybernetics and Mathematics whereas the installed equipment is located mainly at NTNU Gløshaugen in Trondheim. The project's thesis and some of the following results are partially based on a preliminary project conducted by the author during the mandatory subject TTK4550, which acts as an introductory study related to the main masters thesis. The work done in the preliminary project is intended as a basis for the final masters thesis report, completing a 5 year master's degree in cybernetics and robotics at NTNU, Department of Cybernetics(ITK). This report is written to be an independent documentation of the work done during the masters thesis project and on all participating modules described, but a more thorough description of several aspects regarding components, techniques and methods is made available in the bibliography and with appropriate citations throughout the report.

Intended Audience

This report is constructed and written to aid future work with respect to any topic relevant to the content. The report is based on knowledge in the fields of cybernetics and information technology with a correlating vocabulary. The reader is assumed to be competent in basic programming theory, electrical engineering and realtime requirements methodology. Any additional information required to fully understand the implementation and design strategy will be elaborated in a separate chapter.

Prerequisites

This project depends on multiple hardware and software components, mostly embedded modules necessary to complete a functional logging device. An Amazon Web Service (AWS) account is necessary to utilize the services provided by AWS and a computer station with internet access is necessary to manage and upload applications to the cloud.

List of necessary equipment and software

- LPCXpresso4367 development kit
- ESP8266MOD NodeMCU
- Kaifa Nuro AMS MA304H3E
- Digi XBee cellular 3G
- AWS console account (software)
- MCUXpresso IDE (software)
- KiCad (software)
- Oscilloscope
- NXP LPC-Link2 JTAG debugger
- M-Bus transceiver

Acknowledgements

This report is supported by complementary work on cloud service technology and utilization by Marit Tundal[40] and a collaborating work on configuration of the 2G/4G modem and cloud management between the author and Håkon Hansen M.Sc.[26]. Acknowledgments are made to their contribution on wireless communication between host and server devices. Much appreciation is granted to my sister, Ane Elisabeth Grande, for her help on grammar inspection. A sincere appreciation is given to my supervisor, Geir Mathisen, for his initial paper idea and guidance throughout the course of this project. An additional acknowledgement is made to NTNU dep. of Cypernetics and robotics and SINTEF Department of Cybernetics and Mathematics for its continuous supply of advise and counseling on hardware design, production and validation.

Table of Contents

Abstract **i**

Abstract **i**

Preface **ii**

Table of Contents **viii**

List of Tables **x**

List of Figures **xiii**

Abbreviations **xiv**

1 Introduction **1**

1.1 Project Description 4

1.2 Project restrictions 5

2 Background and Literature Review - Establish the Foundation **7**

2.1 Literature Review 8

2.1.1 Traditional Distribution Network Design 8

2.1.2 Key Challenges For Distributed Power Generation Networks 8

2.1.3 Radial network transparency 9

2.1.4 Distribution Networks As A Smart Grid 10

2.1.5 AMS deployment and its benefits 11

2.2 Existing Solutions 12

2.2.1 Dent Instruments, ElitePRO 13
2.2.2 En-Mat Datalogger 13
2.3 Regulations 14

3 Theoretical Foundations And Technical Descriptions 15
3.1 FreeRTOS 16
3.1.1 Configuration of FreeRTOS 16
3.1.2 Tasks 17
3.2 Meter-Bus 18
3.2.1 Overview 18
3.2.2 ISO/OSI hierarchy 18
3.2.3 Physical Layer specifications 19
3.2.4 Hardware Specifications 20
3.2.5 Data Link Layer 21
3.2.6 TSS721A an M-Bus Transceiver 22
3.3 Advanced Metering System (AMS) 24
3.4 Home Area Network - HAN 25
3.5 Device Language Message Specification - DLMS and Companion Specification for Energy Metering - COSEM 26
3.5.1 Naming And Addressing 27
3.6 Object Identification System - OBIS 27
3.7 Interpretation of DLMS/COSEM and OBIS Codes For the Data-Link and MAC Sub-Layer 30
3.7.1 Start and End of Frame 30
3.7.2 Frame Format 30
3.7.3 Destination Address and Source Address 30
3.7.4 Control Field - CF 31
3.7.5 Header Check Sequence - HCS 31
3.7.6 Destination LSAP 31
3.7.7 Source LSAP 31
3.7.8 LLC Quality 31
3.7.9 Frame Check Sequence - FCS 31
3.7.10 The Final Frame Segment 32
3.8 Cloud Service Technologies 34
3.8.1 Amazon Web Services - AWS 34
3.8.2 AWS IoT Core 34
3.8.3 AWS Identity and Access management - IAM 35
3.8.4 AWS DynamoDB 35

3.8.5 AWS Simple Storage Service - S3 36
3.8.6 AWS Elastic Beanstalk 36
3.8.7 Elastic Compute Cloud - EC2 36
3.9 Message Queuing Telemetry Transport - MQTT 36

4 System Design and Specifications 39
4.1 Overview 39
4.2 Hardware Requirements 40
4.3 Software Requirements 41
4.4 Hardware Constraints 41
4.5 Software Constraints 41
4.6 System Specification 42
4.7 Choosing The Hardware 47
4.7.1 The Main Processing Module 47
4.7.2 The WiFi Module 49
4.7.3 The Modem module 50
4.7.4 M-Bus Transceiver 51

5 Implementation 53
5.1 Hardware Implementation 53
5.1.1 Processing Module 53
5.1.2 Meter-Bus Simulator Circuit 54
5.1.3 Meter-Bus Transceiver 58
5.1.4 Modem Module 61
5.1.5 WiFi Module 61
5.1.6 Hardware Setup 62
5.2 Embedded Software Implementation 63
5.2.1 FreeRTOS 63
5.2.2 DLMS/COSEM and OBIS Interpretation 64
5.2.3 Modem Communication 66
5.2.4 System Configuration 66
5.2.5 Main function 69
5.2.6 M-Bus Simulator 70
5.3 Cloud Service Utilities and Resource Implementation 71
5.3.1 EC2 Instance 71
5.3.2 DynamoDB 72
5.3.3 MQTT Handler 72
5.4 Complete Software Design 72

5.5 The Complete System Solution 73

6 Testing Procedure, Debugging and Experimentation 75
6.1 Functional Testing and Verification 76
6.1.1 Main Processing Board 76
6.2 Simulated Environment Verification 77
6.2.1 M-Bus Simulator Circuit Test 77
6.2.2 WiFi/3G Test 77
6.2.3 System Configuration Test 78
6.2.4 MQTT Publication test 78
6.2.5 Cloud Storing Test 78
6.2.6 Complete Application Functionality 78
6.3 Test and Experimentation Setup 79

7 Results 81
7.1 Hardware Assessment Results 81
7.1.1 M-Bus Simulator circuit 82
7.1.2 M-Bus Transceiver 83
7.1.3 LPCXpresso4367 83
7.2 Software Assessment Results 84
7.2.1 System Configuration Results 84
7.2.2 AMS Message Handling Results 86
7.2.3 Wireless Communication Assessment Results 88
7.2.4 Cloud Service Results 88
7.3 Simulated Environment Results 89

8 Discussion 91

9 Conclusion 101

10 Future Work 103
10.1 Propositions for Future Work 103

Bibliography 105

Appendix A 109

List of Tables

3.1 ISO/OSI stack 19
3.2 Telegram Format in M-Bus 22
3.3 Field and bit description in the telegram frame 23
3.4 OBIS groups with their respective description 29
3.5 Multiple frames separated by the Flag field 30
3.6 Frame format field 30
3.7 The **MAC** and **HDLC** frame 32
3.8 Listing of all codes part of a short AMS message 33

4.1 Specifications for the HAN system and corresponding acceptance requirements 44
4.2 Specifications for the main processing system and corresponding acceptance requirements 45
4.3 Specifications for the wireless communication system and corresponding acceptance requirements 46
4.4 LPCXpresso4367 specifications 48
4.5 ESP8266E specifications 49
4.6 Comparison of power consumption for the Digi 3G and the LTE-CAT modem 51

5.1 NPN-PNP transistor characteristics 56
5.2 Modem to LpCXpresso4367 connection pins 61

7.1 M-Bus simulator circuit test values and resulting outputs. The last two results marks the threshold for maximum voltage load before BD136 transistor overheats and shortcuts. 82
7.2 Message reception integrity compared to system baudrate. Valid Baud rates experienced no message errors. 86

List of Figures

2.1 Common design of a distribution network 8
2.2 A principal design of the radial distribution network with the integration of an AMS device for each node in the network. 10
2.3 Estimated number of AMS installations. `https://www.nve.no/Media/5662/ams_status_juni17.pdf` 12
2.4 Dent ElitePro XC, `http://www.ie-central.com/wp-content/uploads/2014/08/ESP-with-antenna-2.png` 13
2.5 En-Mat Premium Data Logger. 14

3.1 FreeRTOS task scheduling 17
3.2 Bit representation of an M-Bus signal 20
3.3 Transmission of a single packet in M-Bus 21
3.4 Diagram of the TS721 24
3.5 A typical AMS device from Kaifa with additional display and human interfacing buttons 25
3.6 Client–server model and communication protocol for DLMS/COSEM 28

4.1 Kaifa-specific OBIS list information 42
4.2 General OBIS list for the HAN specifications in Norway 43
4.3 LPCXpresso4367 development board 48
4.4 The ESP8266MOD module with integrated WiFi 49
4.5 The Digi XBee Cellular 3G Global Modem 50
4.6 TSS721A-based transceiver with galvanic insulation 52

5.1 Schematic representation of the M-Bus simulator circuit 57

5.2 PCB Layout for Meter-Bus simulator circuit 58
5.3 3D image of the M-Bus simulator PCB and the physical device after production. 59
5.4 TSS721A-based transceiver with galvanic insulation 59
5.6 TSS721A- and MAX3232-based transceiver with galvanic insulation 60
5.5 TSS721A-based transceiver without galvanic insulation 60
5.8 ESP8266 NodeMCU 61
5.9 Full visualization of the hardware installation. The WiFi module is connected to the processing board on this figure, while the modem is located to the right, unconnected. 62
5.10 Flow chart of the DLMS interpretation procedure. Notice that this procedure will effectively produce a false output until the 7E stop flag has been detected and the message status is set as *package complete* 65
5.11 OBIS interpretation program flow chart 66
5.12 JSON structure for type 3 list AMS messages 67
5.13 Access point program flow chart 68
5.14 Main execution loop for the ESP8266 69
5.15 The main application execution flow with the FreeRTOS scheduler active 70
5.16 Complete software suite for the project 73
5.17 The complete system solution with hardware and cloud services . 73

6.1 Full system test setup 79
6.2 Full hardware test rig. To the right the WiFi module is seen installed on top of the LPCXpresso4367 board. The left side depicts the AMS simulator as a combination of the M-Bus simulator circuit and the Arduino Nano providing the AMS messages. The middle module is the M-Bus transceiver with optocoupled galvanic insulation. 80

7.1 Screenshot of an oscilloscope measuring the output pins of the M-Bus simulator. Notice the 5V scale of the oscilloscope image. The voltage differentiation is clearly observable to be 13V, determined by the 13V Zener-diode. 82

7.2 Screenshot of an oscilloscope measuring the output from the M-Bus transceiver during simulation. Notice the 2V scale of the oscilloscope image. 83
7.3 Response message from the ESP8266 during UART echo 83
7.4 Response message from the LPCXpresso4367 during UART echo 83
7.5 Response message from LPCXpresso4367 and ESP8266 during dual UART echo 84
7.6 Terminal output during system configuration 84
7.7 Screenshot of the configuration HTML page generated by the WiFi module. 85
7.8 Message reception on the LPCXpresso4367. The messages are displayed as read by the UART handler. 86
7.9 The generated JSON string after DLMS/COSEM and OBIS translation 87
7.10 Debugging serial output after successfully translating an AMS message and initiates MQTT transfer. 87
7.11 Partial image of the generated JSON string for a type 3 message . 88
7.12 Snapshot of the DynamoDB table after a logging period 88
7.13 DynamoDB content plot. The plot is generated in Matlab using the derived values from the data table. 89

10.1 OBIS codes for different meter types 109
10.2 Descriptive list of the OBIS codes for the HAN specifications . . . 110

Abbreviations

AMS	Advanced Metering System
AWS	Amazon Web Service
COSEM	Companion Specification for Energy Metering
DLMS	Device Language Message Specification
DNM	Distribution Network Manager
DSO	Distribution System Operator
EC2	Elastic Compute Cloud
GPIO	General Purpose Input-Output
HAN	Home Area Network
HV	High-Voltage
IoT	Internet of Things
ISO	International Standards Organisation
IEC	International Electrotechnical Commission
IDE	Integrated Development Environment
LV	Low-Voltage
M-Bus	Meter Bus
MCU	Micro Controlling Unit
MV	Medium-Voltage
NEK	Norsk elektroteknisk komitè
NVE	Norsk vassdrag- og energiforbund
OBIS	Object Identification System
OSI	Open System Interconnection
RTC	Real-Time Compliance
SDK	System Development Kit
SOC	System On Chip
S3	Simple Storage Service
TIA	Telecommunications Industry Association
TTL	Transistor-Transistor Logic
UART	Universal Asynchronous Receiver-Transmitter
USART	Universal Synchronous Receiver-Transmitter

Chapter 1

Introduction

The pursuit for renewable and distributed energy generation and high- quality power supply prolonged by a constant increase in demand for electric power has introduced a vast array of new technologies with the intent of expanding decentralized energy generation and counteract degrading power distribution infrastructure. Improvements on already established technologies has allowed the private consumer market to participate in this evolution by enabling intermittent renewable energy generation. Technologies related to photovoltaic and wind energy conversion has rapidly been increasing its efficiency, like the increased photovoltaic conversion factor from 1 percent per 1940's to a staggering 41.1 percent as of today[6]. By installing devices and systems designed to accommodate energy conversion, either from natural resources or human activities, small-scale distributed generation is introducing severe implications to the already degrading distribution network[24]. The immense transformation, from centralized to distributed energy generation, is now becoming a challenge to any distributed system operator, mainly concerning stable voltage levels and predictable power characteristics [29].

Distributed power generation, which has evolved from being a supplementary technology to a primary source of energy, is now partially responsible for obsoleting power grids and their basic designs[20]. These profound implications are partially due to a problem with the scaling factor but also based on the present grid design. Most power grids are designed to allow a sparse set of power plants to generate and provide most of the power available on the distribution network, making it dependant on locally placed transformers and high voltage levels across the long-distance transmission lines. This design has well-known restrictions and flaws but has previously served as an adequate solution.

Allowing distributed, small-scale generation of electric power to enter the grid would contribute to divide the generation load and reduce the cost of maintenance and further extension of large power-plants. This contribution could benefit the consumer, especially considering the leverage of inherent properties of small-scale distributed generation technology. In contrast to diverse power generation technology, the demand for more electric power is expected to rise as more equipment depends on electric power. Electric vehicles and complete heating/cooling systems are about to become common amongst the nominal citizen, further increasing dependencies to high-quality power supply and distribution. If each residence is expected to envelop usage of one or several electric vehicles, the corresponding increase in the supply of electric power is necessary to accommodate appropriate charging availability. Since an ordinary day is subjected to a variable demand for electric power, mostly due to non-homogeneous routines, one can expect the overall demand to peak at specific points during the day[20]. This increased demand may affect the voltage levels to slope downwards along the radial transmission line, resulting in the end node receiving a below-acceptable voltage level. The opposite may also be considered a problem. During a sunny day, solar energy production may generate a peak on the delivered power. As most residents tend to reduce their usage of electric power when nice weather occur, either because more people are spending the day outside or because artificial heating is no longer a necessity, the power grid can suffer a considerable drop in electric power demand, resulting in an undesired high power supply at a given node [10].

A simple solution to counteract a population's pattern of behavior is to write billing policies with the specific goal of forcing routines to change in favor of stability. These measures may be deemed unjust or inefficient by the consumer, but may evidently be proven sufficient. Albeit, as the population grows beyond the limits of the current distribution infrastructure new approaches has to be considered, especially as maintenance and adjustments to the already installed infrastructure is a costly affair. The Norwegian government decided in 2011 to compromise between scaling, innovation, and costs by passing a new regulation [8]. This regulation dictates installations of advanced metering system devices (AMS) mandatory in all domestic buildings by 2019. The regulation and its inherent impact on the monitoring capability is the first step towards a complete ability to measure power levels across the national power distribution network, including the radial low voltage (LV) domain, which has previously been a blind spot for the distribution network operator (DNO) [24]. Each AMS is equipped with a sensor device capable of measuring the power production and consumption of each node. The AMS device is also equipped with a modem for transmitting the information on a proprietary channel to a centralized receiver [24]. This information will aid in mapping consumption of electric power across the whole distribution network and add a potential to act upon differences in consumption on local parts of the network. Transparency on the LV segment could enhance optimization protocols, thereby reducing inefficient use of power, provide a potential reduction in power-grid failure and overall increased electric power quality. Since the AMS device is holding information at each node, the managing institution can also allow a locally generated power supply to enter the power-grid, increasing revenues for the nationally regulated institutions and apply cost reduction to the private subscriber if power generating equipment is installed and operated. The intended AMS is only allowed to send information to the distribution network operator, but as the information could be utilized to enhance the local radial distribution segment, a second device could be installed. The idea is to integrate a data acquisition device at each node on the radial distribution system. Connecting this device to the already installed AMS device and allow for real-time data transfer to the DNO. The data would provide the operator with all information necessary to instantly counteract any disruption or malfunction on the power distribution network. For a centralized controller to be able to adjust the transmission line loads, either to reduce or to increase the active effect, the data acquisition device must be required to deliver data in real-time with high demand for integrity. This monitoring property, in accordance with a suitable regulator, allows for a more stable power-grid during fluctuations in electric power demands, and may, in turn, allow each node to deliver energy to the grid. Such installations could reduce the need for large power-plants to be the sole contributor to electric power generation. Technologies similar to this principle is already being developed and is also partially being deployed [29].

By the regulation issued under [8], no other than the national database for power consumption, ElHub, is allowed access to the data provided by each AMS. This conflicts with the ethics of privacy, but is defended by a secondary communication port present on all AMS devices. This port, the HAN port, provides the user with an interface, capable of delivering all measured data responsible for determining the ultimate cost of power. Data made available through the HAN port is not obtainable without auxiliary equipment designed to be compatible with the data stream and its transmission protocol. With multiple AMS suppliers, each with their own economic interests in mind, custom designs and system architectures which are compatible only with their respective devices, are often the only available auxiliary equipment for the customers, further increasing the cost for each subscriber if such an equipment is desired. In addition, the AMS is currently regulated to only transmit data every ten seconds to the distribution network operators, making real-time regulation inapplicable [1]. To overcome these and other handicaps and insufficient policies, a secondary device could be designed and installed, utilizing the available HAN-port. This device would take advantage of the information, which is made available to the subscriber, and send that data to a cloud service for storage and data analysis. The analysis could then provide a foundation for necessary actions suited to optimize the distribution network characteristics. Real-time data availability would allow each node to be monitored and implicitly regulated according to the power demand in real time and could alleviate in enhancing the transmission-line voltage quality. Such a device would have to make use of the Meter-bus (M-Bus) standard, equipped on all AMS's by the regulation [8]. The collected information could then be filtered, analyzed and sent to the controller for appropriate actions. The information could also be used by the subscriber, directly at each network node, providing real-time monitoring capabilities for the subscriber, aiding in lowering power consumption costs if desirable.

This report focuses mainly on the auxiliary device for data acquisition from the AMS HAN-port output, which could benefit multiple parties in the electric power consumption cycle. The main feature of the device is the acquisition of the data, enhanced with functionality for wireless transmission capabilities and decentralized processing and storage, and is intended for the distribution generation operator. However, the device could be a benefit to any party possessing the AMS device as the data made available could be utilized to raise awareness about local consumption trends and its correlated cost.

1.1 Project Description

This project is intended to investigate the possibilities of implementing and installing an auxiliary data-acquisition device as a supplement to the AMS device, installed in all Norwegian domestic residences by 2019. The device should inhibit logging functionality of all data provided by the AMS and transmit these datapoints to any preferred centralized/decentralized server/cloud service. The data

could then be used to generate electric power consumption trends and enhance the ability to act upon rapid changes in consumption, either rapidly increasing consumption or power production. By providing data from each node in a neighbourhood of residences the low voltage segment of the distribution network could become effectively transparent to the distribution network operator. Allowing a centralized/decentralized server to accommodate a log for each node in a power-grid network would help determine if a distinct node could harm or degenerate the quality of the power transmission line.
The project will provide a specification on constructing a prototype data acquisition device and all necessary specifications for engineering the software suite appropriate for the device hardware. A simulated test will be conducted with a known dataset to assess the functionality of the system and provide a basis for comparing with actual in-field data. An in-field test can be conducted to create a realistic data-set, which could then be analyzed to determine if the dataset is appropriate for an inferential conclusion regarding average electric power consumption.

1.2 Project restrictions

During the development of the project, various restrictions were detected, either from hardware restrictions and flaws or software dependencies. The crucial hardware part, the AMS device, was malfunctioning during the early stages of the project, introducing a severe handicap to the project scope as no real data were available. The project supervisor was unable to provide a replenishment to the malfunctioning AMS, and the choice of finding a workaround was taken. To overcome the problem of missing a data source, an old data-set from the previous project were used as a basis for a simulator capable of providing simulated M-Bus data. This solution added some delays to the project progression and was initially not expected as part of the main scope. Due to the hard deadline of the project, a full-scale test could not be completed within the time limits. Thus, the in-field test was discarded from the scope of the project. During the initial stage of the project, the supervisor planned for a customized hardware device, incorporating all necessary hardware for the module to work. The external institution SINTEF was supposed to provide design and manufacturing of the card, but since no card was provided during a reasonable amount of time this approach was rejected. Lack of a specialized hardware module reduced hardware access to commercially available modules and limited the extension of the project to a single device. This reduction from multiple devices to a single device further decreased the scope of the project to only account for a single prototype, capable of being extended to multiple devices in a future project. However, the project would still investigate and discuss how multiple systems could interact and how the correlating data-set could be used to enhance monitoring capabilities and load regulation.

Chapter 2

Background and Literature Review - Establish the Foundation

Identifying a problem require sufficient information and an established foundation for the current state of the domain. This chapter will dive in to the literature available from research and investigations on the subject of power distribution networks and their current state. Furthermore, this chapter will elaborate how technologies and research has progressed and developed tools, methods and devices for accumulating data provided by smart metering devices. The Chapter will briefly list samples of relevant equipment currently present on the commercial market. Even though this chapter will try to display the general state of the research community, much research is currently under development and the reader should be informed about updated research before making and conclusions based on this chapter. The first section lays the foundation for the problem description and sets most of the conditions for developing the specifications in chapter 4.

2.1 Literature Review

2.1.1 Traditional Distribution Network Design

During the last decades, the standard framework for constructing a general purpose power distribution network has been left unaltered. The principle relies on the top-down approach, meaning that supply of power is mostly provided by large power generation installations and facilities. The power is transported as either high voltage (HV), medium voltage (MV) or low voltage (LV), on separated sections of the distribution network [24, p.9]. These sub-divisions of voltage levels facilitates a varying degree of monitoring capabilities. Typically the HV networks are constructed in a mesh topology with a high level of transparency. These sections are carefully monitored and regulated by the distribution system operator (DSO) to be kept within appropriate levels across the network. This degree of monitoring properties from the sub-transmission levels of HV to the radial and dispersed LV is quickly more frequently based on aggregated measurements at stations linking the different divisions together [24, p.7-9]. Limited network transparency means that the operator is partially left blind and unable to determine if a single residence is responsible for the majority of power consumption along a radial line segment or if the consumption is divided amongst multiple residences.

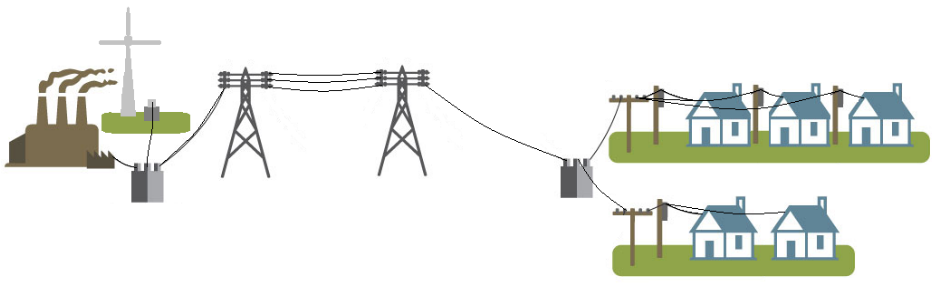


Figure 2.1: Common design of a distribution network

2.1.2 Key Challenges For Distributed Power Generation Networks

The inclusion of distributed power generation to the power distribution network is considered beneficial to all included parties regarding power sufficiency, load division, network degeneration, and cost. The future distribution network is expected to incorporate systems designed to enhance the quality of power, reduction of transmission costs, high supply integrity and less congestion. All of these factors are already heavily investigated to actively increase the functionality of the distribution network or to lower prices [24, p.3-4]. Although these design constraints and principles appear to be a natural evolution of any distribution network, the reality is somewhat diminishing. Regulating distributed production is often managed by the owner of the equipment used for the production and is therefore rarely present in the incorporation of distributed generation systems. Long distances to the load would require the dispatcher to know whenever supply is needed along the local distribution network section to account for fluctuations in demand for power [24, p.6]. Besides, the current voltage-levels obtained through distributed generation systems may not coincide with the voltage levels present in the distribution network transmission line. Thus, adjustments to the transmission load, dispatched voltage levels and distribution timing is necessary to allow distributed generation to enter the distribution network actively[20]. In contrast to the positive stimulation of distributed generation available to the network, several aspects tend to jeopardize current policies regarding distribution network integrity. To be able to adequately accommodate small-scale distributed generation injection to the network, distribution network managers have to consider all possible scenarios related to the integration. If distributed generation systems (DGSs) are present on a LV section the distributions system may be forced beyond the prohibited voltage level due to varying power-demand and power injection. As these thresholds are constructed based on physical properties of the transmission cables and auxiliary hardware, they cannot be allowed to exceed the operating boundaries of the distribution network. Threshold exceedance may actively contribute to congestion on the distribution network, and the injection will have to be discarded. Several investigations reveal that one of the most frequent security threats to any distribution network is the presence of voltage increasing beyond the safety threshold or the allowed current limit threshold [35].

2.1.3 Radial network transparency

The LV segment of the distribution network is labelled the radial segment. This segment extend from the last voltage reduction stage, at the transformer station. This station is the last point for the distribution network operator to monitor voltage levels, power and load before it enters application equipment and regular facilities [35]. Each Transformer station is equipped with a metering unit for monitoring the output provided by the transformer. Without further metering devices along

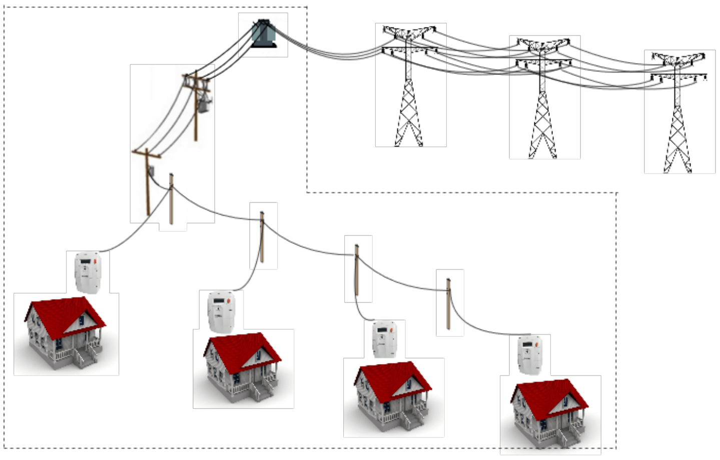


Figure 2.2: A principal design of the radial distribution network with the integration of an AMS device for each node in the network.

the LV transmission line the transformer station is the last node for monitoring power on the distribution. Thus, without any equipment to monitor power along the LV distribution line this segment is considered non-transparent to the DSO. The concealing of power consumption in a neighbourhood through lack of monitoring capabilities handicaps the distribution network with high latency to active power load corrections, which could contribute to congestion in power supply and degrading power supply quality to each node [35]. To account for low transparency and no monitoring of neighbourhood load profiles, each node in a LV distribution network must be equipped with a metering device for power consumption tracking. With the introduction of AMS installment for in-house monitoring, a system of AMS devices would suffice as a source for attacking the lack of transparency. Acquisition and analysis of data from a metering device, located at each node in the LV distribution, could be utilized to create the missing load profile of the neighbourhood. The contribution of a real-time acquired load profile could effectively reduce the high latency power regulation and provide a high-integrity regulation system as part of a fully functional smart-grid architecture. A typical low transparency segment is depicted in figure 2.2.

2.1.4 Distribution Networks As A Smart Grid

As the distribution networks currently progress to be such a fragile piece of the infrastructure a redesign of its vital parts could enhance several properties of the network characteristics. Incorporating extensive voltage monitoring would allow the DSO to obtain a full visual display of the state at which the distribution network currently remains[20]. Sections could be monitored individually, resulting in an opportunity for controllable power injections along the distribution line. Incorporating regulation systems at LV intersection points could further enhance the ability to act upon rapid changes on the transmission lines, suppress violations of threshold levels or counteract congestion. A system of devices intended for these purposes, in cooperation with the distribution network and its operator, defines the smart grid architecture. Facilitating regulation and avoidance of excessive distributed generation injections is presented as a critical property to further evolve the distribution network as proactive infrastructure development for accommodating renewable energy sources as a feasible competitor to traditional energy generation[24].

2.1.5 AMS deployment and its benefits

The advanced metering system (AMS) consists initially of a metering unit and a communication module. These two components provides a foundation for distributed monitoring capabilities, removing blind zones and therefore contributing to transparency on the power distribution network [1]. Transparency is the critical necessity for a smart grid to be constructed with functional integrity. When transparency on all levels of the distribution network is obtained, the DNO is able to obtain information about fluctuations on the grid, which could be used to initiate necessary measures for corroborating stability and optimization of the network. Access to data from each node on the distribution network could further be used for improvement on the energy diagnostics and more detailed network load profiles [20, p.6-31]. AMS installation is becoming mandatory for all residences in Norway by 2019. A prognosis of the installation rate is depicted in figure 2.3. The implications of real-time access to actual power consumption could benefit the consumer as the data is made available through a user-friendly interface. Each AMS is configured with a standardized access port, named; Home Area Network port (HAN) [1]. When activating the HAN port and installing custom or third-party accessories, a user can access the actual data used to create the consumption billing prices for electrical power. Awareness about actual power consumption could help the consumer to regulate how and when electrical equipment is used, which could aid the consumer in reducing consumption during high network load periods. A reduction in the overall network load is the intended objective with AMS installment. However, when transparency on the distribution network is achieved, multiple infrastructure enhancements could be integrated [20, p.32-35]. Distributed generation of power could be an enterprise for private residences, wishing to participate in renewable energy production or merely obtaining revenues from electrical energy generation. The induction of electric power to the distribution network needs to be regulated for power supply quality to be persistent during power induction to the distribution network [20]. Below is a short summary of the benefits of installing AMS devices and allowing the provided data to be monitored by the DNO.

- Reduction in power consumption fluctuations
- Transparency to the radial distribution network
- Detection of power-failure or excessive power leaks
- Consumption awareness and power redistribution
- Real-time notifications to the consumer
- Possibility for actively reducing electrical power costs

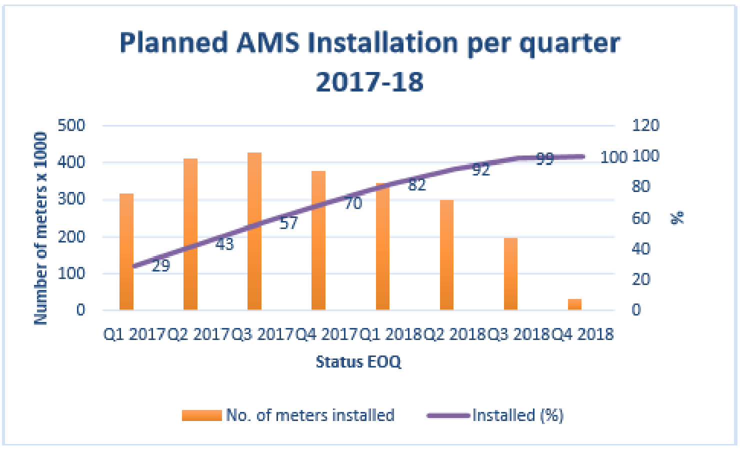


Figure 2.3: Estimated number of AMS installations. `https://www.nve.no/Media/5662/ams_status_juni17.pdf`

2.2 Existing Solutions

This section will briefly list some examples of existing solution per May 2018. The list is just an example of many available commercial products available as data logging devices, capable of connecting wireless to the internet. Most of the devices listed are dependent on a local router to transfer data to the cloud and is not configured explicitly to function in collaboration with the Norwegian standard AMS device.

2.2.1 Dent Instruments, ElitePRO

A well suited equipment for tracking energy consumption is the Dent Instruments ElitePRO data logger[27]. This compact device is designed to allow monitoring of electrical energy and power consumption. The device is capable of measuring over four channels simultaneously on the voltage interval 0-600V AC/DC. These measurements may be sampled either from a single phase or a three phase supply. The device is equipped with an internal memory of 16 MB of non-volatile memory for data storage and a communication interface supporting USB, Bluetooth or WiFi.

Figure 2.4: Dent ElitePro XC, `http://www.ie-central.com/wp-content/uploads/2014/08/ESP-with-antenna-2.png`

2.2.2 En-Mat Datalogger

The ENMAT Premium Data logger is a versatile Internet ready unit supporting 8 Pulse Inputs and 8 Analogue CT Inputs. It also supports Modbus RTU and TCP/IP communications. [23]. The device is integrated with a cloud based software, effectively transferring all data to a cloud server based on a CSV file transfer. It is equipped with a battery backup and time synchronization features. The En-Mat data logger costs about *3000USD*, but includes a subscription to the proprietary cloud service provided by En-Mat.

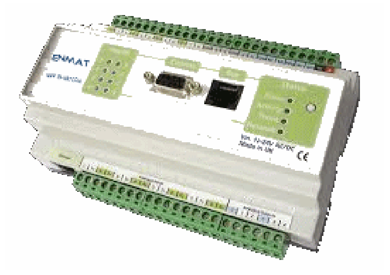


Figure 2.5: En-Mat Premium Data Logger.

2.3 Regulations

The Norwegian government issued a regulation June 24, 2011,[8] stating that all domestic residences should be equipped with an AMS metering device by 2019. This device should be used in accordance with the billing policy to the power distributor for enhanced billing accuracy and stimuli to reduce power load during high consumption periods. The regulation incorporates use of a institutionally regulated database, ElHub, for data storage and processing. The regulation also includes a stipulated requirement for data transmission frequency, format and hardware components.

Chapter 3

Theoretical Foundations And Technical Descriptions

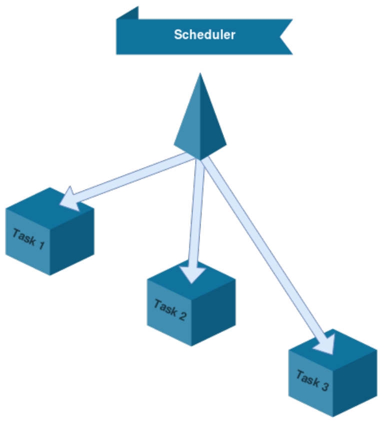
This chapter will present the necessary foundation required to follow methodology and nomenclature used in the upcoming chapters. The chapter will elaborate on and specify technical details related to the meter-bus, FreeRTOS, cloud technologies, modem technologies, and real-time programming theory. The chapter will focus less on common theory and will omit theory and methodology presumed ordinary for the intended audience.

3.1 FreeRTOS

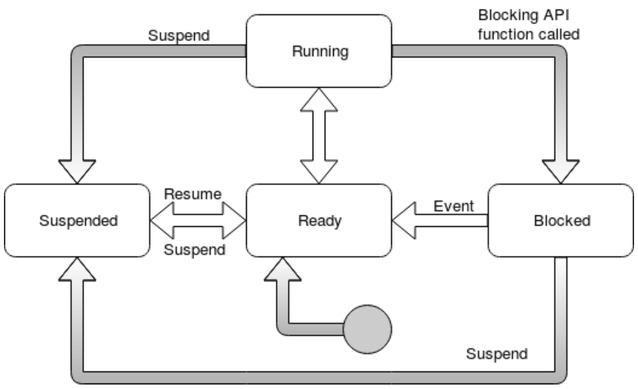
FreeRTOS is a real-time kernel developed to schedule any application requiring real-time demands. Richard Barry initially developed the kernel around 2003 and later maintained and further developed the operating system with his initiated company, Real Time Engineers Ltd.[25] FreeRTOS has been extensively used for 15 years and has now been switching ownership to Amazon Web Services, which also develops a cloud service suited to run interweaved with the operating system[12]. The kernel is designed to be a robust and diverse tool for managing real-time applications on embedded systems, and functions as a quality-controlled operating system without proprietary dependencies. The kernel is also an open source project and does not require a license to install nor modify locally. The OS is designed to be a strict real-time operating system whose primary task is to disallow any non-deterministic operations to be executed during the normal mode of operation. The OS is cross-platform supported and can be mounted on 30 different microcontrollers without any need for customization. FreeRTOS is confined within relatively small libraries, written in the C programming language, containing each function as a separate file. Tasks, queuing, semaphores and timing each have their respective libraries, and the user is free to include the ones necessary to execute their application. The choice of omitting parts of the libraries results in a significantly reduced software footprint, favorable for small embedded systems[25]. One of the paramount properties of the OS is to schedule tasks during run-time. Thus, both the scheduler and its required tasks will be further explained along with some of the real-time tools beneficial to real-time applications.

3.1.1 Configuration of FreeRTOS

The OS can be configured through a designated configuration header file named *FreeRTOSConfig.h*[19]. This single file is the sole contributor to customize FreeRTOS for its intended target. Thus, tailoring the FreeRTOSConfig.h file correctly, and placing it in an appropriate location in the system hierarchy, are important steps to complete before initiating development on the application functionalities, intended to exploit the OS features. Before utilizing any of the OS application programming interfaces (APIs) the FreeRTOS.h file must be included in each API source file. The FreeRTOS.h header file contains all prototypes necessary to make any application run as part of the OS scheduler. The main application function is where the scheduler is initiated.



(a) Three tasks in conjunction with the scheduler



(b) Normal mode scheduler flow chart

Figure 3.1: FreeRTOS task scheduling

3.1.2 Tasks

FreeRTOS is structured around the use of real-time application tasks (Ch.3 in [25]). Each task is independently processed and executed in the real-time scheduler. The task architecture ensures that no coincidental dependencies are necessary for the task to complete its execution during run-time, which makes each task an atomic function-set and does not affect the other tasks 3.1b, except if poorly structured in the scheduler. To ensure that all registers, variables, and values are not affected by a task swap, the FreeRTOS supplies a private part of the stack to each task initiated. This division of RAM implies an increased memory requirement when utilizing tasks in FreeRTOS. Large memory requirements could imply that tasks are not suited for implementation when memory is restricted, and in that case, a co-routine would suffice as a feasible substitution[19]. Co-routines avoid the use of separate stack areas but are prone to misuse if special consideration is not made regarding variables and other dynamic memory-dependent functionalities. Utilizing **Task** features require the task.h file from within the FreeRTOS library. The library needs to be included where tasks are declared and initiated. After a task is declared and defined it has to be initiated using the scheduler. After the initialization in the scheduler, all further management of the task is done by FreeRTOS and its included functionality.

3.2 Meter-Bus

3.2.1 Overview

The Meter-Bus (M-Bus) is a European standard for transmitting process measurements information, such as temperature, pressure, and power. M-Bus is not a network. Hence, it relies mostly on the last two levels in the ISO/OSI standard [34]. M-Bus is designed to be a robust and reliable communication protocol and inhibits noise-resistant properties to ensure deterministic data transfers in challenging environments. Meter-bus was first developed by Dr. Horst Ziegler at the University of Paderborn [3] in cooperation with Texas Instruments [39]. The protocol was designed to meet the requirements of interconnected devices over medium distances, including the presence of electromagnetic disturbance and demand for parasitic power-deduction. The demands for high integrity, low-cost systems able to handle harsh industrial environments led to the development of the Meter-Bus [3].

3.2.2 ISO/OSI hierarchy

Meter-bus is defined as a protocol standard and is not considered a network. The state of the M-Bus as a protocol allows the M-Bus to exclude parts of the ISO/OSI hierarchy stack. The standards included in the protocol corresponds mainly to the physical- and the data-link layer of the hierarchy. Albeit not considered a network, the M-Bus do in fact include parts of the network layer, but correlates this layer with no specific standard. The transport layer, session layer, and presentation layer are all left empty as there is no need for these layers to be specified.
The M-Bus incorporates an extra layer to obtain control over bit parity, baud rate and start/stop bits. This extra layer, the management layer, breaks with the ISO/OSI standard as the management layer is allowed influence across the stack hierarchy as well as being an additional layer to the stack hierarchy. Table 3.1 lists all the layers incorporated into the M-Bus protocol.

Layer	**Features**	**Standard**
Management layer	parameter settings, baud-rate, address, bit parity	-
Application layer	Data structures, data types, actions	EN1434-3
Presentation layer	empty	
Session Layer	empty	-
transport Layer	empty	-
Network layer	Extended addressing	-
Data Link Layer	Transmission parameters, Telegram formats, Addressing, Data integrity, (**Signal requirement**)	IEC 870, (**IEC 7480**)
Physical Layer	Bit representation, Bus extension, Topology, Electrical specifications	M-Bus

Table 3.1: ISO/OSI stack

3.2.3 Physical Layer specifications

The M-Bus protocol is defined as a Master/Slave protocol with half duplex communication controlled by a Central Allocation Logic unit (CAL) [3, ch.4]. The CAL function as the master node whereas the bus slaves typically are sensors and terminals, connected in parallel. The Bus specifies a two-wire bus line between the master and the slaves connected on the bus line. Power is drained from the bus by utilizing the nominal voltage levels in the bit configuration. The M-bus documentation specifies the bit shifts as a **mark** for logic 1 and **space** for logic 0. The **mark** corresponds to a nominal voltage of *+36 V DC* at the driver output. The **space** reduces the nominal voltage by *12 V DC* to a differentiated voltage level of *+24 V DC*. For slave-master communication, a current-modulation is applied. The **mark** is represented as a constant current of *1.5 mA* while the **space** is represented as a current drain requirement at the slave node of *11-20 mA*. The **mark** state is often exploited to power any external sensors (slaves) and the interface itself, by the principle of parasitic power supply.

The **mark** state is used as the M-Bus's quiescent state which corresponds to *+36 V DC* at the master and *1.5 mA* at the slave. The specifications for the physical layer effectively restrict the slave to not exceed *1.5 mA* as a constant quiescent current. The draining of the bus, (i.e. the ability to parasitically power the interface),

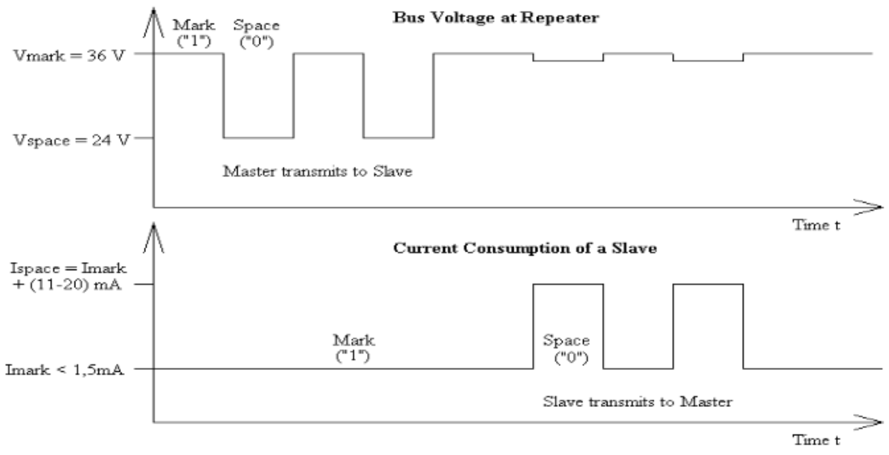


Figure 3.2: Bit representation of an M-Bus signal

is done when no slave is sending a space. This draining will result in a **mark** voltage at the master which is lower than *+36 V DC*. Thus, the node must adjust itself to the actual quiescent current to enable detection of *11-20 mA* increase in current and inhibit the ability to detect a change in *12 V* to detect a space functionally. Due to the double dependency of both voltage and current, the protocol is a half duplex. The high levels of voltage and the current passing through the wires mean that the protocol answers to a high degree of interference resistance.

3.2.4 Hardware Specifications

The M-Bus specifies a two wired standard [3, ch.4]. Thus, the M-Bus slave/master device can easily accommodate off-the-shelf components as its base transmission medium. The specifications limit the maximum distance between repeaters to be less than *350 m* and a total bus length of *1000 m*. The total capacitance of the M-Bus protocol cable medium cannot exceed max *180 nF*. The total number of slaves allowed to meet the requirements for the parameter restrictions are 255. The protocol operates at baud-rates between *300 and 9600 Baud*.

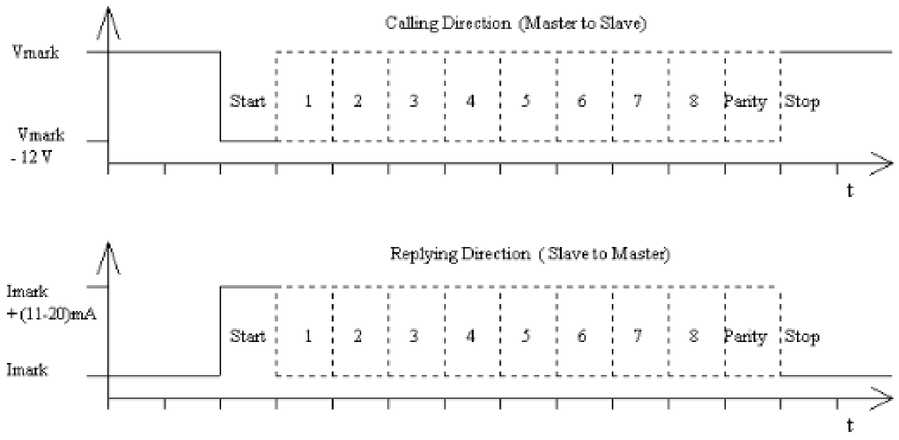

Calling Direction (Master to Slave)
Vmark
Vmark - 12 V
Start
1
2
3
4
5
6
7
8
Parity
Stop
t
Replying Direction (Slave to Master)
Imark +(11-20)mA
Imark

Figure 3.3: Transmission of a single packet in M-Bus

3.2.5 Data Link Layer

The specifications described in the physical layer are partially responsible for regulating the data-link layer of the M-Bus protocol stack [3, ch.5]. The data link layer describes an asynchronous serial transmission with half-duplex properties and a baud-rate of *300 - 9600 Baud*. The data link layer of M-Bus is standardized in IEC 870-5, including the sub-category *Signal Requirements* which are more thoroughly defined in IEC 7480.

The M-Bus data link layer specifies asynchronous serial bit transmission, whereas synchronization is obtained through the first and last bit in a message, the start and stop bit. The start bit has to be a **space** to account for the quiescent current, and concurrently the stop bit has to be a **mark**. The data link packet is composed of a start bit, eight consecutive data bits, the parity bit and the stop bit. A total of eleven bits with the least significant bit as the lowest bit. A packet transmission is performed in the order depicted in figure 3.3.

The data link specifies the telegram formats used for the M-Bus packets. Table 3.2 displays these telegram formats. The telegram formats are categorized into four types: a **Single Character**, a **Short Frame**, a **Control Frame** and a **Long Frame**. The respective descriptions are available in table 3.3. There are two types of transmission services taking place in the data-link layer. A *Send/Confirm ser-*

Single Character	Short Frame	Control Frame	Long Frame
E5h	Start 10H	Start 68H	Start 68H
	C field	L Field = 3	L field
	A field	L Field = 3	L Field
	Check Sum	Start 68H	Start 68H
	Stop 16H	C Field	C Field
		A Field	A Field
		CI Field	CI Field
		Check Sum	User Data
		Stop 16H	(0-252 Bytes)
			Check Sum
			Stop 16H

Table 3.2: Telegram Format in M-Bus

vice and a Request/Respond service. The Send/Confirm service *SND-NKE* serves to start after an interruption or communication initialization. The service require the master to set the *FCV* bit-field to 1 and await the response of the slave. The correct confirmation response of the slave will incorporate the **single character** frame with the content of an *E5* character combination. If the master doesn't receive the acknowledgement frame it resend its own acknowledgement frame until a slave responds. Request/Respond procedure happens when the master requests data from the slaves. The master sends a packet according to the class 2 frame format and waits for a slave to acknowledge with the data frame requested. Communication errors during transmissions are also detectable by the data-link layer. By checking the content of the start/stop bits, checksums and the second start character of the long frame or control frame the master can detected whether all communication is preformed without loss of data or de-synchronized packet exchange.

3.2.6 TSS721A an M-Bus Transceiver

Texas Instruments, one of the collaborators of the M-Bus protocol specifications, have designed a specialized chip for transmitting/receiving an M-Bus signal [28]. The chip, called TSS721A is a small package chip with few prerequisites and a low physical footprint. The component is designed to be as simple as possible, yet completing all of the requirements set by the M-bus specifications. The IC will function as a repeater in the M-Bus network as well as a voltage level converter, effectively makes the M-Bus signal usable to any TTL restricted circuit.

Single Character	An acknowledge receipt of transmission and a single character transmission
Short Frame	Contains two fields, C and A, with included check sum. The start bit is the character 10H and the stop bit is the character 16H
Control Frame	A long frame without user data. Holds predefined control sentences. Encapsulates check sum
Long Frame	Start bit as 68H, stop bit as 16H. Contains three functional fields, A, C and L. Encapsulates a check sum at the end
C Field	The control field
A Field	The address field
CI field	Control information field

Table 3.3: Field and bit description in the telegram frame

The TSS721A operates on *300 - 9600 Baud* and only require *3.3V DC* to operate. The power requirements may be supplied by an external source or may be drawn from the M-bus itself, as the M-Bus is capable of delivering parasitic power to a node on the network. The diagram of the internal circuitry can be seen in figure 3.4

TSS721 Functionality

The TSS721A is equipped with reversed polarity protection, which effectively makes the M-Bus connections interchangeable on the input and output. This property is achieved by demanding the M-Bus signal to pass a rectifying bridge after being pushed through an external protection resistor located outside the IC. The rectifier can be seen in figure 3.4 as BR. To detect the voltage differentiation between the two wires of the M-Bus, the TSS721A utilizes a comparator, revoking a signal affirmation trigger when a +12V DC potential is detected between the two wires. The comparator utilizes an external capacitor SC to adjust the nominal voltage in the comparator to the **mark** voltage level. This capacitor will also enable the TSS721A to operate during a **space** state. The *8.6V* charge stored during a **mark** is then discharged during the **space** state. The TSS721A is ale to provide a

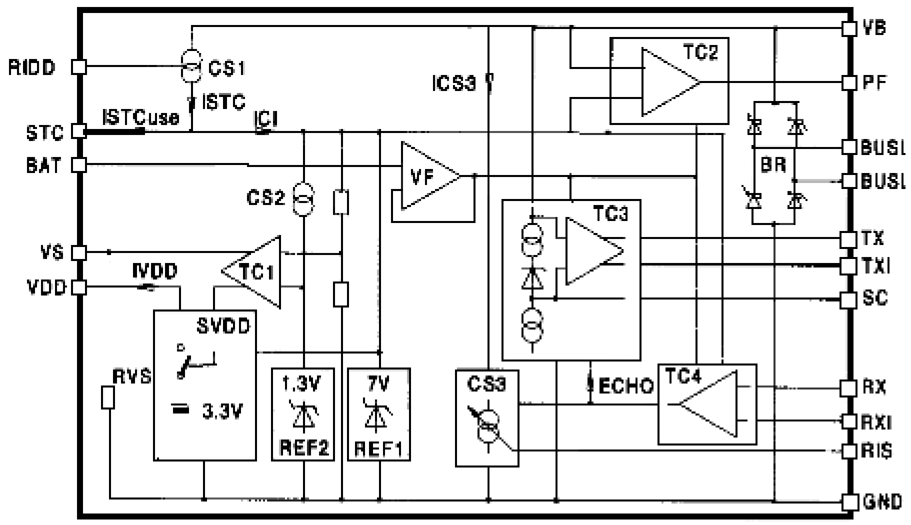


Figure 3.4: Diagram of the TS721

power supply to any externally connected peripheral through its integrated **VDD** pin. The power supply pin can deliver up to *3.3V DC*. If a pulse current requirement is present, the TSS721 utilizes an external reservoir capacitor, holding up to *7V DC*. This potential is then transferred to the VDD pin as *6V DC* during a high state pulse.

3.3 Advanced Metering System (AMS)

The advanced metering system (AMS), sometimes referred to as ”smart meters” is a collection of devices constructed with the intent of gathering information from various analog sources and deliver that information to a particular receiver [43]. The AMS device is usually configured to either measure electric energy or flow characteristics of fluids. Many AMS devices utilize wireless technology for communication with the centralized receiver, whereas modems are the most widely used broadcasting technology. A typical AMS device is confined in a closed shell and is isolated from interacting with its environment. The sensor itself is also guaranteed to be intrinsically safe through galvanic insulation against its connected peripherals. The specifications qualify for high industrial standard equipment and enable the device to be mounted and installed in harsh conditions without risking hazards to other equipment or its users. The AMS design selected to be implemented in Norway follows the regulations stipulated by the Norwegian government, where quality assurance is governed by the Norwegian department of hydroelectric power (NVE) [1]. The specifications include the Meter-Bus (M-Bus) as a secondary protocol for wired communication to an external interface. The M-Bus is available through a designated home area network port (HAN-port), connected with the RJ45 connector standard. Each AMS isolates itself from the high energy power conductor with a galvanic insulator, protecting vital parts of the interior from any harmful energy-peaks potentially occurring on the measured medium.

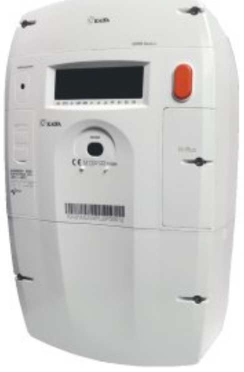


Figure 3.5: A typical AMS device from Kaifa with additional display and human interfacing buttons

3.4 Home Area Network - HAN

Interconnecting different types of devices and entities around the house requires some form of standardized protocol. The digitization of homes is mainly based on ”smart devices”, which implies the capability to function without intervention from an operator. The smart feature of many devices is based on self-evaluation and system restoration, internal data processing, error handling and automated communication with an external platform, in overall devices with ubiquitous properties [5]. To enable the device and all other smart devices to be interconnected for sharing information the devices need to be attached to a seamless interface network with predefined specifications. The interface network responsible for inter-connectivity between metering and automation devices in a typical home is known as the Home Area Network (HAN). A HAN comprises several smart devices, often smart metering devices set to the task of monitoring consumption and similar digitization procedures [32]. In conjunction with these smart metering devices one may often find one or several servers, usually to process data and provide the user with graphical or numerical representation. The server could either be a cloud-based application or a localized server with a user interface. To keep track of data submitted to the network the DLMS/COSEM with OBIS is often integrated into the devices in the network. The DLMS/COSEM model function as an abstract description of the device and provides classes of data-values suited to complement the measuring units of the metering device. The integration of DLMS/COSEM and OBIS is quite substantial and will be elaborated in 3.5. Simple and cheap attachment equipment is desired in order to minimize complexity and costs related to installing a new device to the network. Thus, a typical wired Ethernet or a WiFi connection will be adequate and sufficient for establishing network connection between the entities in the network. The HAN is mainly constrained by the perimeter of the actual building in which it is confined, but with a connection to a router or modem, the network is capable of transmitting data beyond the boundaries of the house and utilize the internet as well.

3.5 Device Language Message Specification - DLMS and Companion Specification for Energy Metering - COSEM

The device language message specification (DLMS) is an object model and a concept for abstract modelling of communication entities. The DLMS concept works as a way to view functionalities of metering devices from the perspective of an interface[41, p. 14]. DLMS, in collaboration with the companion specification for energy metering (COSEM), follows a three-step approach to define classes and rules for targeted metering devices, capable of communicating with a third party communication entity. The three steps are roughly divided into:

- Modelling
- Messaging
- Transporting

DLMS uses the OSI/ISO model to model information to enable exchange of information between nodes. Thus, and application layer is necessary to handle the concept of information exchange and define rules for communication between entities. It is based on a server/client model with the metering device as the designated server[42, p. 22]. The key characteristics of data exchange using DLMS/COSEM are:

- Clients and third parties may access the metering device.
- Access control is provided through the mechanisms defined in the DLMS/-COSEM AL and the COSEM objects.
- Low overhead and efficiency is ensured by various mechanisms including selective access, compact encoding and compression.
- Cryptographical methods are provided to ensure secure communication.
- Various communication media can be used to connect to HANs, WANs or LANs.
- Data exchange may take place either remotely or locally. Depending on the capabilities of the metering device.

Application functions of metering devices and data collection systems are modelled by application processes (APs) and is configured to work in correlation with the application entities (AE) [42, p. 22]. Each AE represents a single AP, whereas an AP may represent multiple AEs. By specifications, the server APs and the clients are located on separate devices with a communication protocol stack to entangle data exchange between the nodes. The different locations, data flows and protocol stack is visualized in figure 3.6.

3.5.1 Naming And Addressing

In DLMS, naming and addressing are highly binding as the name identifies the entity and is mapped to a specific address, that is, where the device can be found. Each entity in the DLMS/COSEM model is uniquely identified by its name, set by the *system title*, and shall be permanently defined. The address of an entity shall be set according to the type of communication protocol (phone number, MAC-address, IP etc.) used and is either pre-configured or assigned during the registration process.

3.6 Object Identification System - OBIS

Object identification system (OBIS) composes a set of identification codes for equipment compliant with DLMS/COSEM. See section 3.5. Each COSEM entity

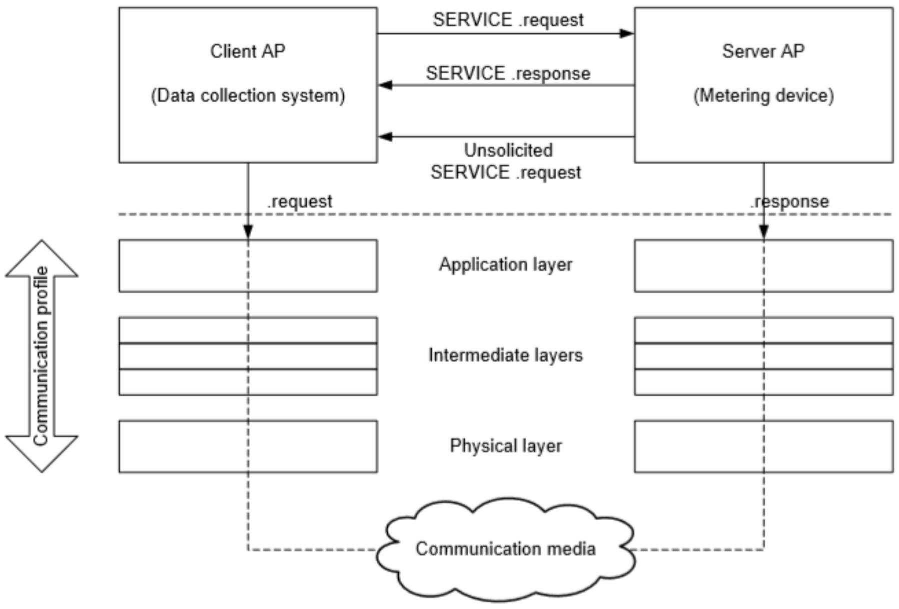


Figure 3.6: Client–server model and communication protocol for DLMS/COSEM

is inherently identified by its corresponding logical OBIS code. Thus, the OBIS codes define the identification codes for commonly used data items in metering equipment[41, p. 142-148]. These items may typically be power, current, pressure or volume measured by the metering entity. OBIS allows the communication interface to correlate measured values with their respective unit of measurement by providing a unique identifier for all data within the metering equipment. The OBIS codes may also include abstract values like identification, address and metering state. OBIS codes are divided into six different groups, where each group contains specific target categories. These groups are displayed in Table 3.4 [41, p. 142]. To each measuring device there is a corresponding OBIS list with all six groups included. Each group may be represented with multiple instances and can be entity-specific for a given producer of the device. This ambiguous standard require each producer to define their respective OBIS table for each device utilizing a distinct OBIS-based service.

Group	Use of group
A	Identifies the media (energy type) to which the metering is related. Data without the status as media related information is handled as abstract data.
B	Identifies the measurement channel number. This allows data from different sources to be identified. Group B may also identify the communication channel, and in some cases other elements. The definitions for this value group are independent from value group A.
C	Identifies abstract or physical data items related to the information sources. The definitions depend on the value in group A. Further processing, classification and storage methods are defined by value groups D, E and F. For abstract data, value groups D to F provide further classification of data identified by value groups A to C.
D	Identifies types and results from physical quantity processing. These values are identified by values in group A and C, according to various specific algorithms (Beyond the scope of this report).
E	Identifies processing or classification of quantities identified by values in value groups A to D.
F	Identifies historical data values. These are identified by values in value groups A to E, according to predefined billing periods.

Table 3.4: OBIS groups with their respective description

Flag	**Frame i**	**Flag**	**Frame i+1**	**Flag**	**Frame i+2**	**Flag**

Table 3.5: Multiple frames separated by the Flag field

Format Type					Frame length sub-field										
1	0	1	0	S	L	L	L	L	L	L	L	L	L	L	L

Table 3.6: Frame format field

3.7 Interpretation of DLMS/COSEM and OBIS Codes For the Data-Link and MAC Sub-Layer

The AMS devices follows the MAC-based sub-layer protocol of the DLMS/COSEM classes which inhibit 10 distinct frame fields.

3.7.1 Start and End of Frame

The Frame type 3 opening flag, based on EN62056-46, sets the start of a frame. The length of the flag field is one byte and is predefined to be the value 0x7E. When two or more frames are transmitted continuously, a single flag is used as both the closing flag of one frame and the opening flag of the next frame. This is displayed in Table 3.5

3.7.2 Frame Format

The frame format consists of three sub-fields with a total of two bytes. It consists of the tree sub-fields:

- Format type sub-field (4 bit).
- Segmentation bit (S, 1 bit).
- Frame length sub-field (11 bit).

3.7.3 Destination Address and Source Address

The **LLC** sublayer contains two address bytes. One for the destination and one for the source.

3.7.4 Control Field - CF

The length of the control field is one byte. It indicates the type of commands or responses, and contains sequence numbers, where appropriate

3.7.5 Header Check Sequence - HCS

The length of the **HCS** field is two bytes. This check sequence is applied to only the header, the bits between the opening flag sequence and the header check sequence. Frames that do not have an information field or have an empty information field do not contain an **HCS** and **FCS**, only an **FCS**. The **HCS** is calculated in the same way as the **FCS**.

3.7.6 Destination LSAP

The **Destination least significant** bit is used for broadcasting features. This value will always be *0xE6*.

3.7.7 Source LSAP

The **Source least significant bit** is used as a command/response and will always have the identifier. value of the Source LSAP is *0xE6* or *0xE7*. When set to 0, it identifies a **command** and when set to 1 it identifies a **response**;

3.7.8 LLC Quality

The Control byte is referred to as the **LLC Quality** parameter. Its a reserved parameter for potential future use. Its value is administered by the **DLMS UA** and currently, it must be set to *0x00*;

3.7.9 Frame Check Sequence - FCS

The length of the **FCS** field is two bytes. Unless otherwise noted, the frame checking sequence is calculated for the entire length of the frame, excluding the opening flag, the **FCS** and any start and stop.

Start/ Stop Flag	Frame For-mat	Destin. Addr.	Source Addr.	Control	Header check se-quence	Info	Frame check squence
1 byte	2 byte	1 byte	1 byte	2 byte	1 byte	X byte	2 byte

Table 3.7: The **MAC** and **HDLC** frame

3.7.10 The Final Frame Segment

Each of the eight previous subsections comprises the **DLMS/COSEM** frame. Combining all elements from the DLMS section one can create a dictionary for a random AMS message. An example of such a dictionary is displayed in Table 3.8.

Code	Description	Value
7E	Frame start/Stop flag	Start
A0 27	Frame format type (4 + 11 bits)	0xA = Type 3, 0x27 = 39 bytes
01	Destination address	0 (terminated by LSB = 1)
02 01	Source address	2 (terminated by LSB = 1)
10	Control field	10
5A 87	Header check sequence	calculated based on address byte
E6	Destination LSAP	Predefined E6
E7	Source LSAP	Predefined E7
00	LLC Quality	Reserved
0F 40 00 00	Information (Carries MSDU)	32 bits
09	Redundant DLMS code	-
0C	Length of String (OBIS)	12 Bytes
07 E2	Year	2018
05	Month	May
10	Date	17
03	Day	Wednesday
12	Hour	18
13	Minute	19
2B	Second	43
FF	Unspecified	Unspecified
80	Unspecified	Unspecified
00 00	Unspecified	Unspecified
02 01	Clock sync	clock check sum
06	Integer next up	integer
00	Byte 1	...
00	Byte 2	...
05	Byte 3	...
28	Byte 4	Byte 1+2+3+4 = 1380W
B8 0C	Frame check sequence	Calculated checksum
7E	Frame start/stop	Stop

Table 3.8: Listing of all codes part of a short AMS message

3.8 Cloud Service Technologies

Cloud storage and cloud-based computing differentiate itself from local computing technologies in the ability to scale up to an almost unlimited size if needed. Distributed computation implies substantial access to computing power with a seamless collaboration between the computing hardware and the storage facility. Gaining access to such technologies requires the resources to be localized on network-connected servers. These distributed chains of servers are called clouds, and their properties and possible applications detain a contribution to any internet of things (IoT) device. Cloud service technologies usually include both the storage of data and access to processing of that stored data. Cloud services are becoming a frequent and accessible solution for handling big data and processing of those datasets. Thus, approaching a problem with storage of data and remote access to that data, a commercially available cloud solution would suffice for most applications.

3.8.1 Amazon Web Services - AWS

Amazon web service (AWS) is a cloud service provider with an incorporated register of multiple cloud computing services[14]. With a total of 100 different services, AWS ranks as one of the most prominent actors in the cloud computing market. AWS uses decentralized or distributed IT infrastructure to make several resources available to customers on demand. AWS provides a flexible and elastic cloud solution with high reliability and cost-effective billing models.

3.8.2 AWS IoT Core

A part of the AWS register of services is the AWS IoT Core[16]. The primary job of the AWS IoT Core is to allow small Internet-connected devices to connect to the AWS service register and channel any communication to other services provided by AWS. The IoT Core is responsible for setting up a secure and reliable channel into the AWS server park, which could then allow data to be stored in databases, provide data to the connected device or alleviate in processing load on the device. The IoT core is comprised of seven features, each responsible for a designated task. The AWS IoT Core Device SDK provides a way to easily connect a device to the IoT Core application.

- The SDK contains internal features for handling secure connection, message exchange, authentication and more.

- the device gateway, a feature serving as the entry point for AWS IoT connected devices. The gateway manages all connected devices and implement semantics for multiple protocols to ensure secure and efficient communication.
- The message transceiver, a low latency MQTT broker for devices to subscribe or publish messages, providing low latency and flexibility.
- Authentication and authorization is managed with mutual authentication between the device and the AWS service. The feature includes x.509 certificates and SigV4 authentication protocols.
- The registry establishes an identity for the connected devices and tracks meta-data, such as attributes and capabilities. Each device is assigned a unique, consistently formatted identification number.
- The device shadow creates a persistent, virtual version of the device connected to the IoT Core. The shadow contains the current state of the device and lets the user monitor the device from within the AWS console.
- The Rules Engine makes it possible to build IoT applications that gather, process, analyze and act on data generated by connected devices at global scale without having to manage any infrastructure.

3.8.3 AWS Identity and Access management - IAM

The AWS Identity and Access Management (AWS IAM) enables the user to manage access to AWS services and resources [15]. The IAM lets the user assign keys and policies for the connected devices and applications. To enable particular devices to access a data table or database the permission rule, role and user credentials must all be configured using the IAM console. The policies of IAM allows the cloud manager to determine how a device accesses services provided by AWS.

3.8.4 AWS DynamoDB

DynamoDB is the AWSs answer to a NoSQL database [17]. The service provides storage of multiple data types and automatic scaling of capacity. Reliability is granted through storage redundancy as data selected for storage in the database is duplicated to three independent locations. The duplication will also reduce access latency and high data integrity. AWS provides monitoring capability to the DynamoDB database, giving the manager high levels of database management.

3.8.5 AWS Simple Storage Service - S3

Amazon S3 is an object storage service provided by AWS[13, p. 1]. S3 generates a table for storing data as objects. These objects contain the stored data, meta-data and a unique identifier. This form of storage allows for storing and retrieving large amounts of unstructured data [13, p. 3]. Data stored in S3 is located in designated, virtual buckets, hence the name S3 Bucket. In these buckets, distinct objects are assigned their unique **keys**.

3.8.6 AWS Elastic Beanstalk

AWS Elastic Beanstalk is an AWS service for efficient application deployment and management in the AWS cloud[18]. It allows for relaxed and non-restrictive scaling of the storage and database services used, and the user only pays for the usage of underlying AWS services.

3.8.7 Elastic Compute Cloud - EC2

Amazon EC2 is a cloud computing service [11, p. 1]. The service is available as a computing capacity linked to a single or multiple *instances*, virtual computing environment or machine located on a centralized/decentralized server. Processing power, memory, storage and network capacity can be configured specifically for each instance [11, p. 1]. The instance can boot as many different operating systems or computing environments, i.e., Ubuntu, Arch Linux and more. To secure the communication between an EC2 instance and a remote user, public-key cryptography is applied to a key-pair[11, p. 508]. Key pairs used for access can be created through the EC2 console or a command line. The terminal of EC2 instances can be accessed remotely by using an SSH client. The approach of using a third-party application requires the acquisition of a *.pem* key file, generated by AWS. This *.pem*-file is then required to be converted to a terminal-compliant *.pkk*-file.

3.9 Message Queuing Telemetry Transport - MQTT

The message queuing telemetry transport protocol is a lightweight publish/subscribe messaging protocol designed for small embedded systems or lightweight environment applications [4]. The protocol is easy to handle and leaves an especially small data footprint when integrated into a software application. The MQTT protocol utilizes a message broker, a server routine for handling incoming messages. Each subscribing device will connect to the broker and request subscription to a designated topic. A topic is a key value representing the publishers set of data values. The protocol follows the ISO standard and is set on top of the TCP/IP protocol, effectively makes it a part of the session layer and application layer.

Chapter 4

System Design and Specifications

This chapter is intended to illustrate and specify the key characteristics of the system hardware and software. The chapter will initiate with a simple overview of the whole system and further list specifications and requirements as a guide for the design and implementation. The list containing the specifications will be cited and used extensively throughout the rest of the report with appropriate citations for each conforming implementation and testing criteria.

4.1 Overview

The complete system module is composed of several sub-modules fulfilling a dedicated task within the complete system. As described in the introduction 1 the system relies on a functional AMS device capable of providing the data stream through its integrated HAN interface. This ability require the module to successfully receive and interpret the data provided by a two-wire differential signal, formatted on the DLMS/COSEM standard with OBIS codes as the identification standard. To accommodate this ability the module needs to possess an M-Bus transceiver and inherent interpretation software integrated on the system core. To possess a means for communicating the data to an external receiver without requiring adjustments to the installation area or compromising security measures a modem, preferably 2G/4G, needs to be integrated. This completes the list of all necessary components to accomplish the project objective.

- Main processing unit (Hardware)

- HAN/M-Bus transceiver (Hardware)
- 2G/4G modem or WiFi (Hardware)
- Main real time OS application (Software)
- HAN/M-Bus driver (Software)
- Modem/WiFi driver (Software)
- Configuration for HAN/M-Bus and modem (Software)

4.2 Hardware Requirements

To enable an M-Bus signal to be read and processed by an external processing unit, the unit will require auxiliary hardware equipped with a communication interface. As the M-Bus protocol exceeds nominal voltage levels respective to transistor logic a conversion from the M-Bus signal to a TTL signal is required for the processor to identify data within the signal. To allow interaction with a cloud service a wireless module is necessary. Utilizing wireless technology instead of wired technology removes the demand for unnecessary cabling hardware and allows the system to be incorporated into confined areas without conflicting with galvanic insulation or isolation. The system will be constructed around a modem or a WiFi module for wireless communication and a development kit for housing the processing unit. The development board must include connection ports for connecting any peripheral hardware deemed necessary to accomplish project objectives. Powering the system will require an external power supply. This power supply will be chosen with respect to the power requirements of the total system. A backup battery could be necessary to account for any loss of power during normal operation. This auxiliary battery will aid the system when the main power supply is below sufficient levels or absent.

Logging AMS data requires possession of an AMS metering device connected to the three-phase power grid. The AMS device is required to follow the standardized template for Norwegian installations, implying the incorporation of the HAN-port. Usually, these HAN-ports are configured for a HAN interface in conjunction with the RJ45 connector, to be the primary connection point, but two-wire access is a sufficient condition for extracting AMS messages.

To be able to convert M-Bus signals to TTL signals the TSS721A[28] transceiver chip will be used in company with the necessary supportive circuitry. The transceiver module will be the single module responsible for lowering the nominal voltage level of the M-Bus signal down to TTL voltage levels.

4.3 Software Requirements

Handling multiple tasks in real time requires a handler and sufficient processing performance. To accommodate real-time properties in the system an operating system is preferred. Selecting an OS is determined by availability, cost, and OS properties. To develop, compile, debug and upload the software package an appropriate compiler and SDK are necessary. Since the LPC43xx series are used to develop and integrate most software during this project, the MCUXpresso IDE is suitable for managing the software packages and configure the files for installation on the hardware. In addition to the main IDE, the modem or WiFi module may require additional software for configuration and installation.

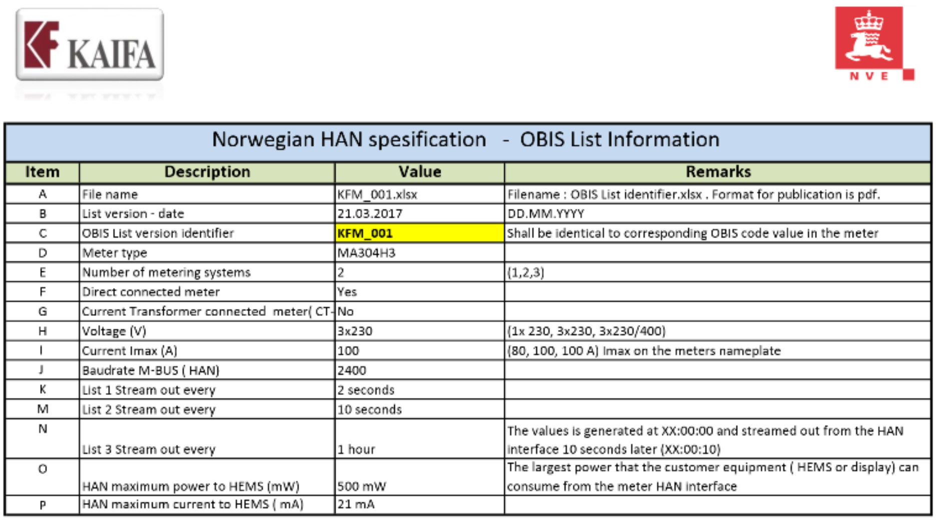
The system software suite needs to be highly modularized for high code quality. The system needs to be divided into several files and will be depended on multiple additional libraries, mostly gnu-cpp compatible libraries. Any deviation from standard open source software will be explicitly commented in the software package description.

4.4 Hardware Constraints

Due to the entitlement as a prototype development project, it follows a strict hardware budget. Thus, all hardware equipment is kept in low quantities and selected with the total cost of purchase in mind. This restriction orients the development process to include only low-cost hardware resources, which further afflicts the choice of controller, additional components, and the wireless transmission module. No custom processing card will be designed to rule out hardware faults and high costs.

4.5 Software Constraints

The software suite development in the project is highly entangled with the choice of IDE selected for writing code. The choice of which microcontroller will also determine how the code is created and compiled. The most traditional microcontrollers usually incorporates the AVR architecture and is well suited for the gcc compiler. The choice of using the gcc compiler restricts all embedded code to be



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	21.03.2017	DD.MM.YYYY
C	OBIS List version identifier	**KFM_001**	Shall be identical to corresponding OBIS code value in the meter
D	Meter type	MA304H3	
E	Number of metering systems	2	(1,2,3)
F	Direct connected meter	Yes	
G	Current Transformer connected meter(CT-	No	
H	Voltage (V)	3x230	(1x 230, 3x230, 3x230/400)
I	Current Imax (A)	100	(80, 100, 100 A) Imax on the meters nameplate
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	

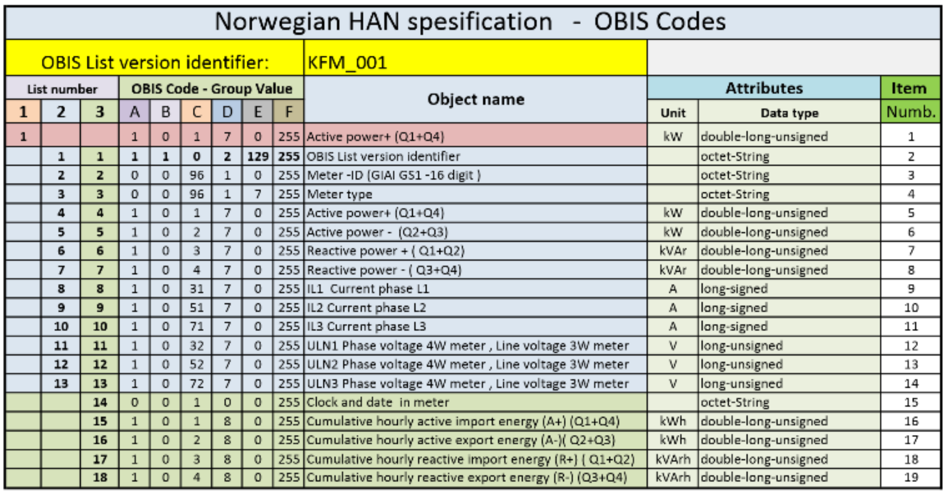
Figure 4.1: Kaifa-specific OBIS list information

either C or C++. An AWS EC2 instance will suffice for a virtual machine to hold multi-language scripts off-location which will account for the environment in the cloud. These scripts may be written in python or any other preferred language.

Implementation of the DLMS/COSEM/-OBIS messages requires information regarding the specific OBIS codes used by the meter and the structure of the DLMS messages. This constraint implies the use of a table containing the codes. The tables used by Norway and Kaifa respectively are displayed in figure 4.1,4.2, 10.1 and 10.2. As the tables illustrate, the Kaifa table will determine how the interpretation software is designed. List 4.1 and 4.2 are sufficient to implement a message interpreter, whereas 10.1 and 10.2 ill act as supplementary documentation and is illustrated in Appendix A.

4.6 System Specification

To ensure that each part of the system operates with adequate performance and no functionality is missed during development a set of specifications is defined and listed. Each specification is ordered by its dependency, starting with the overall specification and splits down into several subordinate specifications. This section presents all specifications necessary to fulfill the project goal. The list is composed of four elements, a specification code for reference during design, implementation and testing and connected acceptance criteria for each specification. The chapter 7 will refer extensively to the acceptance codes to evaluate whether the system operates within the given specifications.



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	Data type	Numb.
1			1	0	1	7	0	255	Active power+ (Q1+Q4)	kW	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)	kW	double-long-unsigned	5
	5	5	1	0	2	7	0	255	Active power - (Q2+Q3)	kW	double-long-unsigned	6
	6	6	1	0	3	7	0	255	Reactive power + (Q1+Q2)	kVAr	double-long-unsigned	7
	7	7	1	0	4	7	0	255	Reactive power - (Q3+Q4)	kVAr	double-long-unsigned	8
	8	8	1	0	31	7	0	255	IL1 Current phase L1	A	long-signed	9
	9	9	1	0	51	7	0	255	IL2 Current phase L2	A	long-signed	10
	10	10	1	0	71	7	0	255	IL3 Current phase L3	A	long-signed	11
	11	11	1	0	32	7	0	255	ULN1 Phase voltage 4W meter , Line voltage 3W meter	V	long-unsigned	12
	12	12	1	0	52	7	0	255	ULN2 Phase voltage 4W meter , Line voltage 3W meter	V	long-unsigned	13
	13	13	1	0	72	7	0	255	ULN3 Phase voltage 4W meter , Line voltage 3W meter	V	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)	kWh	double-long-unsigned	16
		16	1	0	2	8	0	255	Cumulative hourly active export energy (A-)(Q2+Q3)	kWh	double-long-unsigned	17
		17	1	0	3	8	0	255	Cumulative hourly reactive import energy (R+) (Q1+Q2)	kVArh	double-long-unsigned	18
		18	1	0	4	8	0	255	Cumulative hourly reactive export energy (R-) (Q3+Q4)	kVArh	double-long-unsigned	19

Figure 4.2: General OBIS list for the HAN specifications in Norway

Spec. ref.	**System req.**	**Acceptance req.**	**Test ref.**
SR-1	The system should be able to communicate with an M-Bus driven auxiliary system.	M-Bus data is available and convertible to TTL level signals.	TR-1
SR-1.1	M-Bus signal voltage levels must be converted down to TTL voltage levels.	Signal input to the processing unit is between *0 - +5 V DC*.	TR-1.1
SR-1.1.1	The M-Bus transceiver will utilize the *TSS721A* transceiver.	An integrated interface module utilizing the *TSS721A* is connected to the processing unit.	TR-1.1.1
SR-1.1.2	Alternatively to SR-1.1.1 the interface can use the *MAX3232* as an M-Bus transceiver.	The *MAX3232* converts M-Bus to *RS-232*.	TR-1.1.2
SR-1.1.3	If SR-1.1.2 is used the main module must incorporate an *RS-232* to *TTL* converter or an additional circuitry is necessary.	*RS-232* is converted to TTL levels.	TR-1.1.3
SR-1.2	Any M-bus transceiver or converter should not need external power supply.	No external power supply attachment is required for the M-Bus interface to operate.	TR-1.2
SR-1.2.1	The interface module may utilize the parasitic power provided by the M-Bus.	The converter module operates purely on parasitic power.	TR-1.2.1
SR-1.2.2	Alternatively to SR-1.2.1: The M-Bus interface may utilize power supply available through the main controller circuit.	The interface module operates by internal system power supply.	TR-1.2.2
SR-1.3	The system must be able to translate messages provided by the AMS.	Messages transferred via HAN/M-Bus are translated to human readable messages.	TR-1.3

Table 4.1: Specifications for the HAN system and corresponding acceptance requirements

SR-1.3.1	DLMS/COSEM with OBIS translation is performed on-system.	Messages from HAN-M-Bus are translated to be further processed on-system.	TR-1.3.1
SR-1.3.2	Data contained in the HAN/M-Bus message are available for insertion into a JSON structure.	Data content from AMS are available as a JSON structure.	TR-1.3.2
SR-2	The system should contain a main processing module with a dedicated MCU.	The system is capable of running and executing necessary integrated software.	TR-2
SR-2.1	The processing module should contain at least a single CPU.	The system software is constructed to run on a single core CPU.	TR-2.1
SR-2.1.1	The CPU should be able to handle baudrates between *300 - 9600 Baud.*	M-Bus baudrates are supported.	TR-2.1.1
SR-2.1.2	The CPU should be able to handle baudrates between *9600 - 115200 Baud.*	Optimal UART/USART baudrates are supported.	TR-2.1.2
SR-2.1.3	The CPU should be able to handle multiple interrupt sources.	The CPU is capable of receiving and transmitting on separate interrupt UARTs/USARTs.	TR-2.1.3
SR-2.1.4	The register size must be at least 16-bit.	Dual byte sized registers are available.	TR-2.1.4
SR-2.2	The processing board is required to inhibit multiple serial communication standards.	Multiple devices could be connected to the processing module via alternative protocols.	TR-2.2
SR-2.2.1	UARTs/USARTs I/O are available for debugging and handling M-Bus messages simultaneously.	Multiple UARTs/USARTs I/O are present and available.	TR-2.2.1
SR-2.2.2	The processing board is equipped with SPI and CAN to facilitate unique communication protocols with a modem/WiFi module.	A modem/WiFi module can communicate with the processing board via SPI or CAN.	TR-2.2.2

Table 4.2: Specifications for the main processing system and corresponding acceptance requirements

SR-2.3	Power supply is available through jack, USB or two-wire input.	Simple and cheap power supply alternatives are available for the main board.	TR-2.3
SR-2.3.1	USB power could supply the processing module.	A simple USB cable supplies the processing module sufficiently.	TR-2.3.1
SR-2.3.2	A battery backup is easily attached.	The processing module inhibit *5V/500mA* input pins and ground pins for Power supply.	TR-2.3.2
SR-2.4	An operating system is required.	The main application is managed by an operating system.	TR-2.4
SR-3	The system should inhibit a way to communicate data wireless to an off-location server.	The system is able to communicate data to an off-location server.	TR-3
SR-3.1	A modem or WiFi solution should be incorporated into the system.	Long distance, wireless communication technology can transfer data to an off-location server.	TR-3.1
SR-3.1.1	The modem/WiFi module should be able to handle baudrates between *300 - 115200 Baud*.	Interconnection transfer speeds between modem and processor are adequately high.	TR-3.1.1
SR-3.1.2	The modem module can handle UART/USART communication.	The modem is capable of communicating with the processor via UART/USART.	TR-3.1.2
SR-3.2	The modem/WiFi module does not require external power supply.	The processing module can power the modem/WiFi module.	TR-3.2
SR-3.3	The modem/WiFi module should be able to connect securely to a cloud service.	Data transfer from the system can be securely retrieved in a cloud service.	TR-3.3
SR-3.4	System configuration is available through a designated web application.	Users can configure the system via a web-application and log-in feature.	TR-3.4

Table 4.3: Specifications for the wireless communication system and corresponding acceptance requirements

4.7 Choosing The Hardware

4.7.1 The Main Processing Module

Selecting the main processing module for a project can be a vital choice when working with real-time compliant systems and multiple communication channels and protocols. The features of a development board should fit all specifications required to achieve adequate results after implementation. Based on the specifications in table 4.2, the processing board needs to inhibit at least two UART ports with hardware interrupt triggers. The system will also be required to hold a processor with enough throughput to handle the processing speeds required to execute the application with low latency and high integrity. The register size of the cache memory of the MCU will determine how the implementation of the software is arranged, which will further influence the efficiency of the application. A final characteristic to notice is the cost of the hardware. This is a low budget prototyping project and highly priced, low commodity hardware is not desired. All these requirements and prerequisites are listed below.

1. Low cost and high commodity.
2. Multiple UART/USART- and GPIO-ports.
3. Adequate processing speed and register size.
4. Sufficient memory.
5. Common MCU architecture.
6. Good documentation.

Based on this list, the *LPCXpresso4367 development kit* with a *LPC4367JET100 Cortex m4* processor was selected as the main processing module[9]. The project supervisor was responsible for choosing the LPCXpresso4367, equipped with the Cortex M4 chip architecture series. The choice was mainly defended by the board being an efficient, low-cost system with good documentation and being suitable for many different applications and development projects. The LPCXpresso4367 is a versatile and powerful development platform with many different features, including all standard communication protocols (UART/USART (FTDI), SPI, I2C, CAN, I2S, USB, Ethernet (LAN8720A) and 8-pin Segger J-Link JTAG).

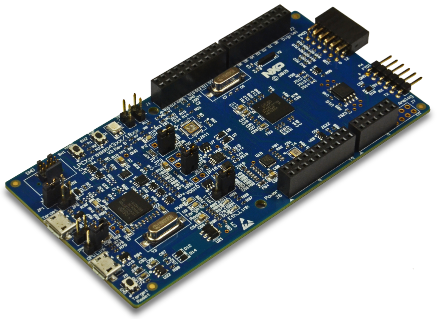


Figure 4.3: LPCXpresso4367 development board

Clock frequency(Hz)	Memory (bytes)	Power consumption	Register size
204 M	1M Flash, 16K EEPROM, 136K SRAM, 64K ROM	80mA, 3.6V nominal	32-bit

Table 4.4: LPCXpresso4367 specifications

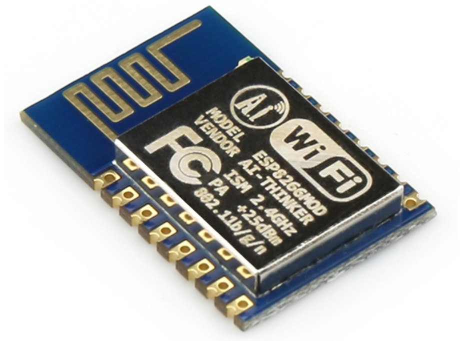


Figure 4.4: The ESP8266MOD module with integrated WiFi

Frequency	Memory	Operating voltage	Operating current	Packet size
2.4-2.5GHz	4Mb Flash	3.0 - 3.6 V	80mA	2.4x1.8 cm

Table 4.5: ESP8266E specifications

4.7.2 The WiFi Module

To test whether the application potentially could be mounted on a much smaller device the ESP8266MOD, mounted on an LPCXpresso4367 compatible shield, will be used as a secondary main processing module. Its integrated WiFi capabilities mainly defend the choice of using the ESP8266 NodeMCU. With a dedicated processor and an integrated WiFi chip, the ESP8266MOD can be used as the WiFi module as well as a standalone processing module. The combination of an IC processor and a WiFi chip makes the ESP8266 a compact and versatile hardware equipment for prototyping medium sized projects. The ESP8266MOD is a self contained System on chip (SOC) with an integrated TCP/IP protocol stack and WiFi direct (P2P). The device is especially cost-effective, in the range of *10-30 NOK* for a single unit[38].

- 802.11 b/g/n.
- 10 bit ADC.
- 32-bit MCU architecture.

- 2.4 GHz WiFi, direct soft- access-point(AP), WPA/WPA2.
- Integrated TCP/IP stack, TR switch, power amplifier, PLLs, DCXO and power management units.
- 19.5dBm output power in 802.11b mode.
- Power doen leakage current ¡10uA
- SDIO1.1/2.0, SPI, I2C, GPIO, PWM, 2xUART, STBC, 1x1MIMO, 2x1MIMO.
- Wakeup- and transmit latency ¡2ms.
- Standby power consumption ¡1.0mW.

4.7.3 The Modem module

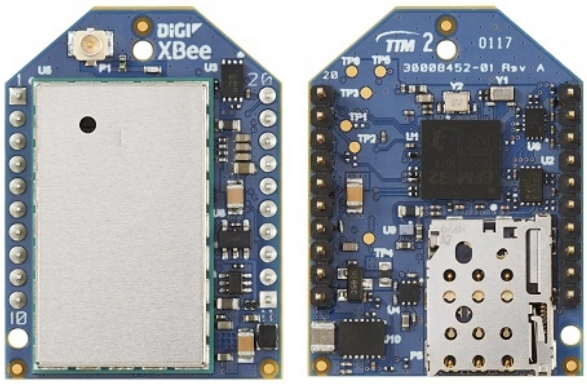


Figure 4.5: The Digi XBee Cellular 3G Global Modem

Selecting the modem module was a joint decision between the author, the supervisor of this project and the author of a different, but similar project[26]. Due to the low project budget, the modem availability was split between the two projects. The Digi XBee Cellular 3G[21] was selected based on its low price, high connectivity, compatibility to the geographical region and its available documentation. The modem supports connectivity via UART and SPI and it operates based on a custom API software designed by Digi and their partner AWS[14]. Another alternative fitting the requirements for the modem is the LTE-CAT[22] version of the Digi

Operation Mode	**3G**	**LTECAT**
Transmit	702mA , 3.3V 425mA , 5V	Avg. 860mA , 3.3V Max. 1020mA , 3.3V
Receive	224mA , 3.3V 160mA , 5V	Avg. 530mA , 3.3V
Idle/Listening	87mA , 3.3V 73mA , 5V	143mA , 3.3V
Deep sleep	10uA , 3.3V	Approx. 10uA 3.3V

Table 4.6: Comparison of power consumption for the Digi 3G and the LTECAT modem

XBee. The two versions are compared in table 4.6, based on the power consumption. The final decision was defended by the power consumption and the pricing of the selected version. The cheaper version was the 3G modem (*709 NOK*[31]), albeit the LTE-CAT version was not considerably more expensive (*787 NOK*[30]).

4.7.4 M-Bus Transceiver

The M-Bus transceiver is a critical component in the system as it is the only protection between the *+36 V DC* M-Bus and the vulnerable TTL level CPU. Galvanic insulation would provide a guaranteed maximum voltage level of no more than *+5V DC* to be fed into the UART ports on the CPU. Thus, an interface with incorporated galvanic insulation is desired. The chosen M-Bus transceiver is an unregistered board bought online. The transceiver is based on the TSS721A and features two optocouplers, one for each communicating direction. The transceiver require parasitic power on the M-Bus side and a supply of *5V DC* with common signal ground on the TTL side. The circuit features reception trigger and a transmission trigger led indicator.

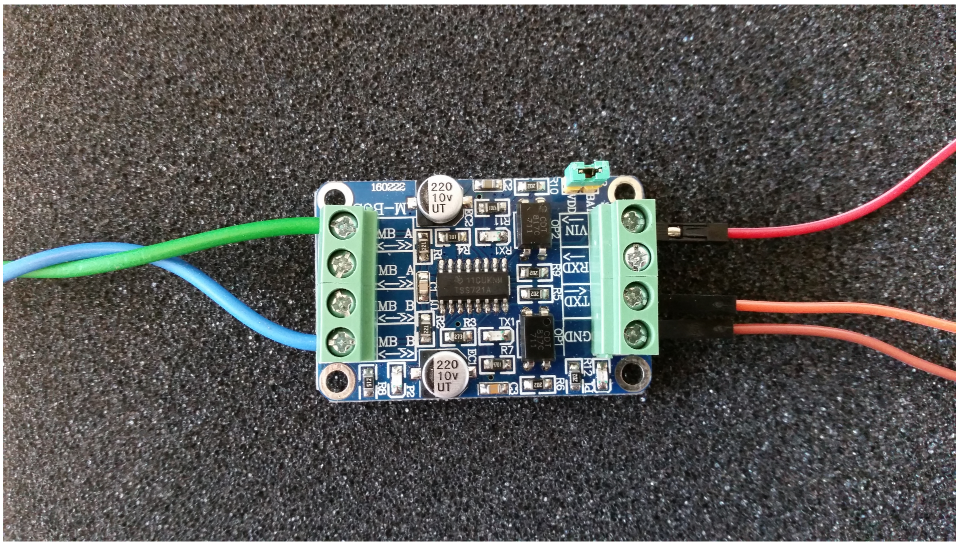


Figure 4.6: TSS721A-based transceiver with galvanic insulation

Chapter 5

Implementation

This chapter will function as documentation for the data acquisition module, constructed to collect and communicate the data provided by the AMS device. The chapter will carefully state any restrictions and requirements for the hardware and software before indulging in any specific implementation strategies or techniques. The implementation chapter will not specify a step by step approach but will elaborate on all individual modules separately and a final assembly of all modules until a seamless system is completed. An additional component, an M-Bus simulator will be described. Due to changes in the project scope this component is not part of the original system specification.

5.1 Hardware Implementation

5.1.1 Processing Module

The processing module is based on the NXP43xx Cortex m4 microprocessor series and fitted on the LPCXpresso4367 development kit, as mentioned in 4.7. The board is powered by a micro USB power supply of *5V DC, 500mA* and incorporates a variety of peripheral connectivity options. The only consideration to the main processing board is to place the USB cable at the right port. The LPCXpresso4367 inhibits two USB micro ports. One for powering the target and the other for combined power and debugging mode. The development kit is also equipped with a socket for an 8-pin JTAG real-time debugger. Utilizing the JTAG removes occupation of one of the UART port available on the LPC4367JET100 chip during debug-mode. Thus, using a JTAG will enable the system to be in debug-mode while utilizing the interrupt triggers on USART0 and USART2.

5.1.2 Meter-Bus Simulator Circuit

The meter-bus simulator will act as a Meter-Bus master node with the sole purpose of providing M-Bus signals to the system for testing and debugging, which will allow the system to be constructed without the need for a functioning AMS metering device installed in proximity to the system during development. To avoid complex architectures and additional sources of errors the simulator is constructed using only ”simple” components. A small **Arduino Nano** will alleviate in constructing the base signal necessary to achieve the switching of the transistors in the circuit. The signal will be oriented to behave like an ordinary AMS message, including the DLMS/COSEM instances and the M-Bus package structure described in section 3.2. The auxiliary circuit will function as a signal amplifier of the provided signal and raise the nominal voltage level of the signal to *+36V DC*. A combination of resistors, NPN transistors, a diode and a PNP transistor will be used to achieve signal alternation with help from an external power supply of *+36V DC*. The Arduino Nano does require a separate power supply of *+5V DC* in which a USB mini cable is sufficient. The circuit power supply has to be able to deliver approximately *40V DC* and *1.5A* to generate an M-Bus signal appropriate for the project. See table 4.2.

Specifications and Design

The M-Bus specifications for master transmission at the physical layer[3] is rigidly defined to be an alternating voltage signal with a voltage differentiation of *12V DC*. To create the base signal a signal generator is necessary. The signal source will be the Arduino Nano controller, fitted with a software package to generate the DLMS/COSEM and OBIS message. The signal generator software is predefined to be output on pin D2 on the controller, but all digital pins and communication pins can be used as output. The circuit is therefore integrated with a designated hub for mounting the Arduino and connect the D2 pin to the rest of the circuit. To switch between output pins the circuit includes a **Source Select** pin, installed with a short-circuit bridge jumper. Secondly, a short-circuit bridge is installed to allow an external source to be delivering the message while the Arduino Nano is installed and active. The *Select pins* and the *External Source* pin are all connected in series, effectively overwriting each other if all sources are active at the same time. Thus, the user must ensure that the bridges are present if an external source is preferable. The external power supply is attached via a two-pin screw-hub for the *+VCC* and the *GND* respectively. The signal will propagate through two *10 kOhm* resistors placed parallel to each other. The resistors are attached to the *base-pin* on transistor Q1 and Q2 respectively, as seen from the schematics 5.1. The signal will trig the flow through each of those transistor, effectively controlling the high voltage flow from external power supply. The Q2 transistor is also responsible for triggering the Q3 transistor. Positioning Q2 and Q3 in series will allow the signal to create an alternating high voltage signal into the BD136 PNP-transistor. With the requirement of a differentiated signal of *12V DC* at the output, a *13V DC* Zener-diode is responsible for guaranteeing *13 V DC* differentiation between the **mark** and **space** of the signal. The Zener-diode is a better choice opposed to the ordinary diode due to its ability to generate a shorter trigger flank. To protect the two M-Bus output ports of the circuit a *220KOhm* resistor is placed between them. The circuit is also design with an additional TTL signal output, purely for testing and debugging. After all components are placed and their value calculated the circuit is simulated in **Spice**, a KiCad plugin software for simulating physical properties of the circuit.

	Collector-emitter voltage	Collector-base voltage	Emitter-base	Collector current	Base Current
BC337	45 Vdc	50 Vdc	5.0 Vdc	800 mAdc	-
BD136	45 Vdc	45 Vdc	5.0 Vdc	1.5 Adc	0.5 Adc

Table 5.1: NPN-PNP transistor characteristics

Schematic

The schematic of the M-Bus simulator is designed in KiCad [2], a free CAD software for designing circuits and their corresponding PCB layouts. Each of the four PNP-transistors is of the type: *BC337* [36], a common PNP-transistor with suitable characteristics for the voltage levels required to produce an accurate representation of the M-Bus signal. The NPN-transistor is of the type: *BD136* [37], a high DC current gain transistor suitable for medium power circuits.

PCB layout

To produce the M-Bus simulator the circuit is drawn in the PCB outline tool. This will effectively enable generation of the Gerber-files necessary to obtain printed PCB boards. The layout is specified with hole mounts, channel width, diameters of the vias and many other parameter configurations.

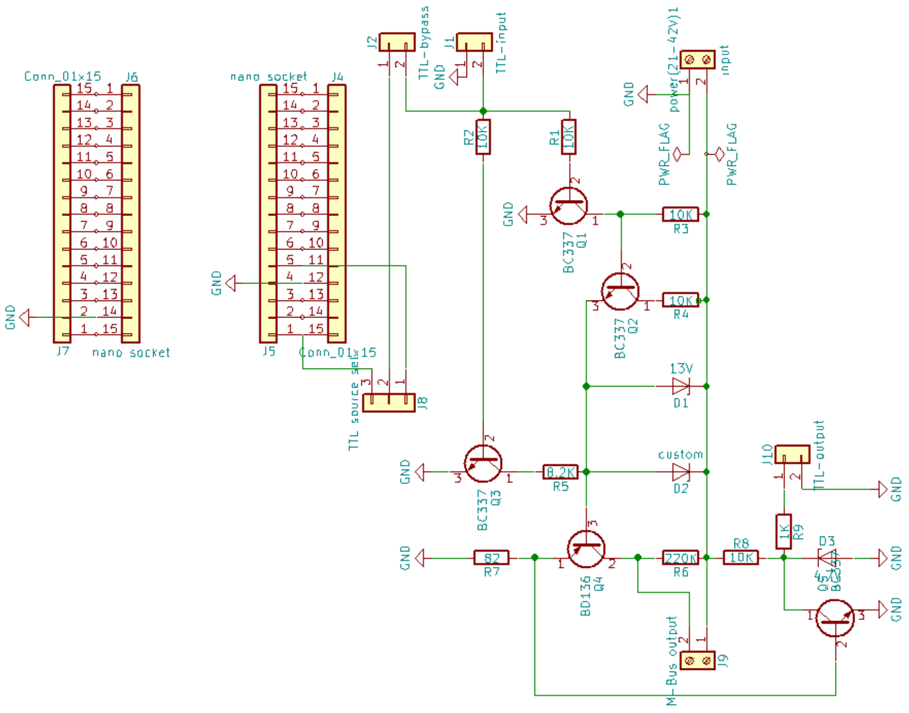
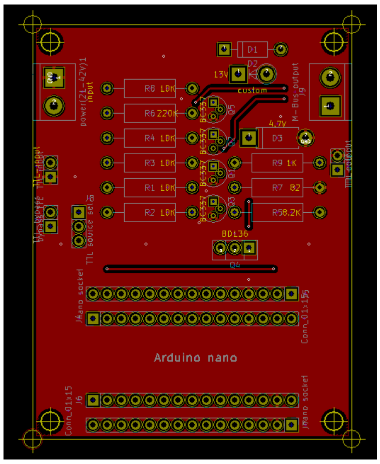
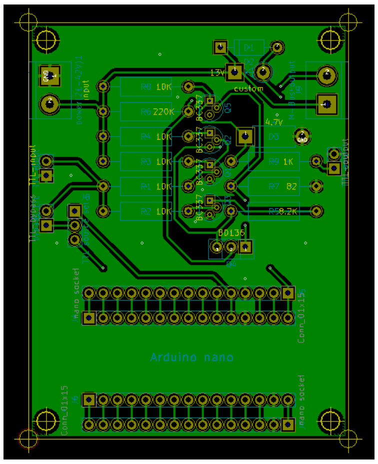


Figure 5.1: Schematic representation of the M-Bus simulator circuit



(**a**) Front copper layer of the M-Bus simulator PCB



(**b**) Back copper layer of the M-Bus simulator PCB

Figure 5.2: PCB Layout for Meter-Bus simulator circuit

The Produced M-Bus Simulator

The production of the M-Bus simulator pcb is performed by the external company *allpcb.com* a pcb production company, providing a variety of prototyping services to a reasonable cost. The montage of the circuit is performed locally using the facilities of *Omega-Verkstedet* `www.omegav.com` at NTNU. After soldering each respective component to its designated spot the circuit is cleaned and tested. The completed production of the board is displayed in figure 5.3. The figure also includes the mounted Arduino Nano, selected to provide the circuit with the designated TTL signal.

5.1.3 Meter-Bus Transceiver

The M-Bus transceiver is designed around the TSS721A chip, and three different designs versions will be used during the project. One with galvanic insulated M-Bus to TTL, one without galvanic insulated M-Bus to TTL and one with galvanic insulated M-Bus to RS-232. The TSS721A transceiver with galvanic insulation utilizes two optocouplers to insulate the attached microcontroller from the high voltage segment of the transceiver. These two optocouplers are driven by the 5V supply from the main processing board. Thus, a guaranteed maximum output voltage from the transceiver is 5V DC. The Transceiver with optocouplers can be

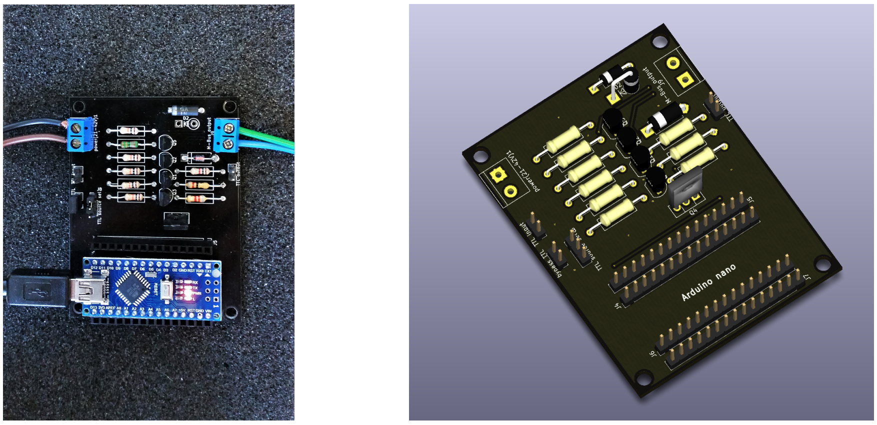


Figure 5.3: 3D image of the M-Bus simulator PCB and the physical device after production.

viewed in figure 5.4.

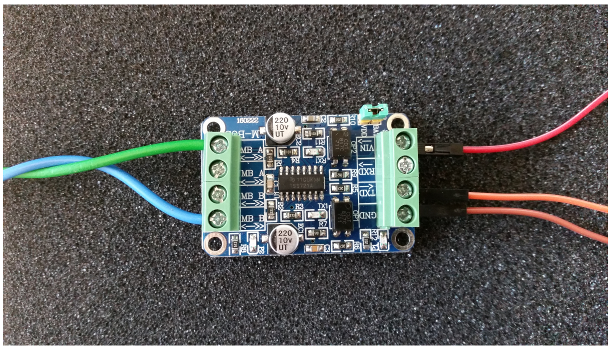


Figure 5.4: TSS721A-based transceiver with galvanic insulation

To simplify the conversion between M-Bus signals and TTL-level signals a converter without optocouplers can also be used as the transceiver. Figure 5.5 illustrates a custom design, again based on the TSS721A transceiver, but with no protection against the potential *+36 V DC* provided by the M-Bus master. This solution is much simpler to implement, but compromises between simplicity and security.

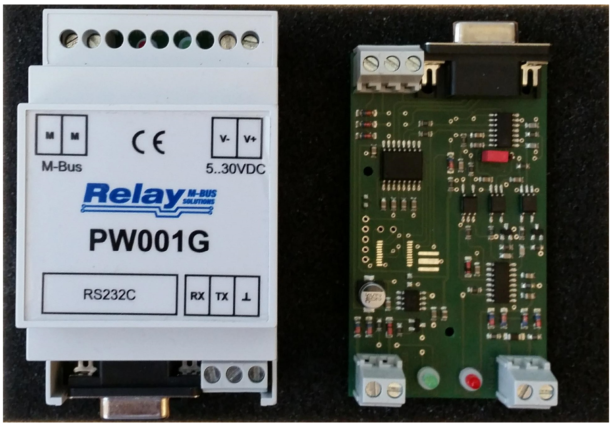


Figure 5.6: TSS721A- and MAX3232-based transceiver with galvanic insulation

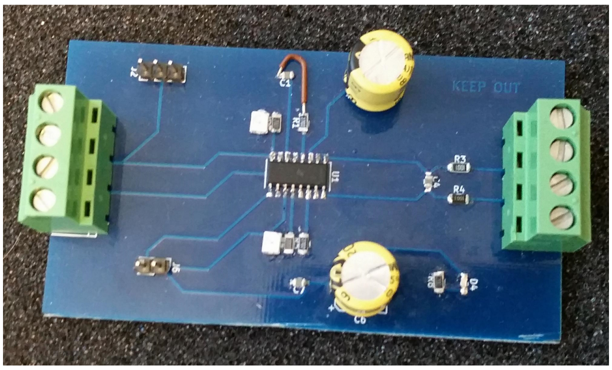


Figure 5.5: TSS721A-based transceiver without galvanic insulation

The final variant can be viewed in figure 5.6. This is a complete transceiver from Relay, a manufacturer of a variety of electrical and digital equipment. The PW001G is an M-Bus to *RS-232* converter with galvanic insulation. The device is primarily used to verify correct signal coming from the M-Bus simulator. The connection to the bus is equal to that of the other two, with both bus-wires connected to their separate port. Connection to the embedded system is not performed in this project since the much smaller M-Bus to TTL with optocouplers is sufficient and avoids the need for an additional conversion from *RS-232* to TTL. The PW001G transceiver is connected to a desktop computer using a serial to USB cable.

5.1.4 Modem Module

The 3G modem is connected to the LPCXpresso4367 using the three necessary UART communication pins:

DigiXBee Pin	LPCXpresso4367 Pin
2 (TX)	J2 18 (RX)
3 (RX)	J2 17 (TX)
10 (GND)	J6 16 (GND)

Table 5.2: Modem to LpCXpresso4367 connection pins

5.1.5 WiFi Module

The Wifi module is implemented as two different hardware components. The testing is done on the ESP8266 NodeMcu development board, whereas the integration to the LPCXpresso is performed using the standalone ESP8266MOD attached to an LPCXpresso-compliant extension board, designed to hold the ESP8266MOD.

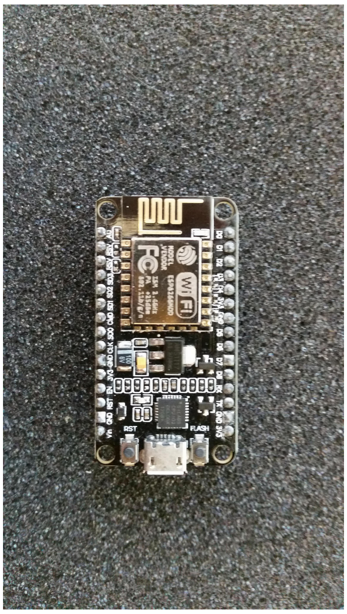


Figure 5.8: ESP8266 NodeMCU

To connect the ESP8266 to the LPCXpresso the extension board is placed as

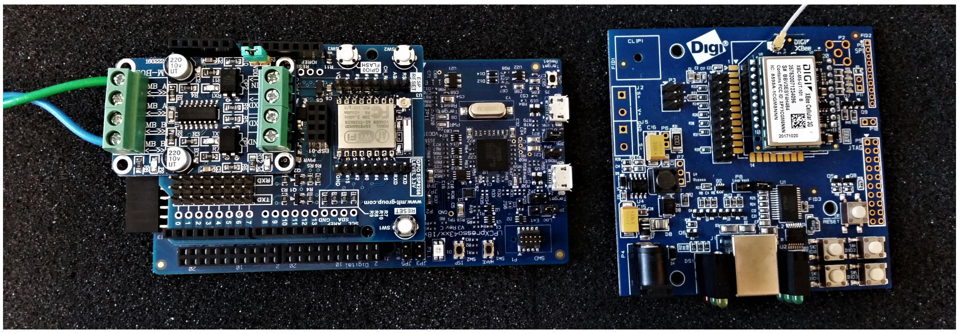


Figure 5.9: Full visualization of the hardware installation. The WiFi module is connected to the processing board on this figure, while the modem is located to the right, unconnected.

displayed to the left in figure 5.9. The extension board fits only in one orientation mapping the utility-pins of the LPCXpresso4367 into the ESP8266.

5.1.6 Hardware Setup

The WiFi module and the 3G modem are tested separately. Thus, they are both connected to pin TX and RX on pin section J2 on the LPCXpresso4367. The Connection to the HAN interface is placed on the TX1 and RX2 on pin section J4. These two correspond to USART2 and USART0 respectively, in the LPC43xx pin mapping firmware. The M-Bus transceiver is connected to the 5V DC output and ground on the main processing board, and the corresponding RX and TX pins are connected to TX1 and RX1 respectively. On HV side of the transceiver, the two signal wires from the M-Bus are each connected to the A and B input. Orientation is irrelevant as the M-Bus is polarity independent. No power is necessary at the M-Bus side of the transceiver since the TSS721A utilize parasitic power supply from the bus-line. The Final attachment is the M-Bus simulator, which is powered by the external power supply of *+40 V DC* and a *+5V DC* power supply to the Arduino Nano. The output gate is connected to the bus-lines coming from the transceiver, again orientation is irrelevant. All preceding sections complete the hardware setup, which can be seen in figure 5.9

5.2 Embedded Software Implementation

This section will describe the implementation of all software packages and scripts necessary to create a functional data acquisition device. The section will elaborate on the implementation of the OS, the necessary drivers and APIs and all scripts located in the cloud service, running on an EC2 Ubuntu instance.

5.2.1 FreeRTOS

FreeRTOS will function as the main operating system, and occupy the sole responsibility for handling all tasks and executions in the main application. Each major software functionality at the main processing module will be enclosed within a separate task and handled according to static priority and queuing policy. The OS will be integrated with an as low footprint as possible to reduce memory occupation and system complexity but will incorporate all necessary functionality to fulfill all specifications listed in table 4.6.

HAN UART Task

The HAN-UART-TASK is the main task for reading and interpreting AMS data acquired by the system. The task is responsible for calling any utility function necessary to translate the raw data message acquired from the AMS device into a human-readable message. To avoid interruption during a critical operation, the task is suited to the lowest priority available, explicitly denying any execution while inside a critical section. The task is active while the ring buffer contains any data and after an interrupt reception from the AMS device.

MODEM/WiFi UART Task

MODEM-UART-TASK is the main task for reception and transmission of data via the integrated 3G-modem or WiFi. The ask is based on the separate API driver for the modem and some utility functions.

Message Data Extraction

This task is responsible for extracting the data stored in the received AMS message. After the HAN-UART-TASK has completed its DLMS translation of the message, the data is still left for interpretation and extraction. This task executes on the verified message buffer and assigns each message category into a JSON structure.

The categories are determined based on a predefined message type list, specific to the meter type in use. These lists are stored in a separate header file named *Kaifa.h*

5.2.2 DLMS/COSEM and OBIS Interpretation

To obtain a human-readable representation of the data provided by the AMS a DLMS interpreter is necessary. This software module is inherently connected to the HAN driver, composing all DLMS/COSEM translations and masks out each category of the message. The categories are predefined in a list, specified for a single AMS device type. Only the Kaifa MA304h3E is supported with a complete DLMS and OBIS list, but any AMS device could be supported with an appropriate software patch.

The DLMS interpretation program extracts one byte at a time from the UART ring buffer. The program checks to see whether the extracted code matches the specifications, available in chapter 3. The procedure of extracting and comparing with the specifications is performed until the stop flag is detected. This indicates that the next section of the message is the OBIS data section. The meter-specific DLMS information is preserved until the JSON object is constructed. Once the OBIS interpretation is initiated, the different list version types are filled with their respective value from the message data section. Each OBIS category is extracted using the same ”extract a byte and compare”-procedure. The difference is whenever values containing multiple bytes are extracted. These cases are handled with a special *getIntVal*-function, effectively accounting for multiple bytes per OBIS code. The DLMS/COSEM interpretation procedure is available in figure 5.10 whereas the OBIS interpretation procedure is available in figure 5.11.

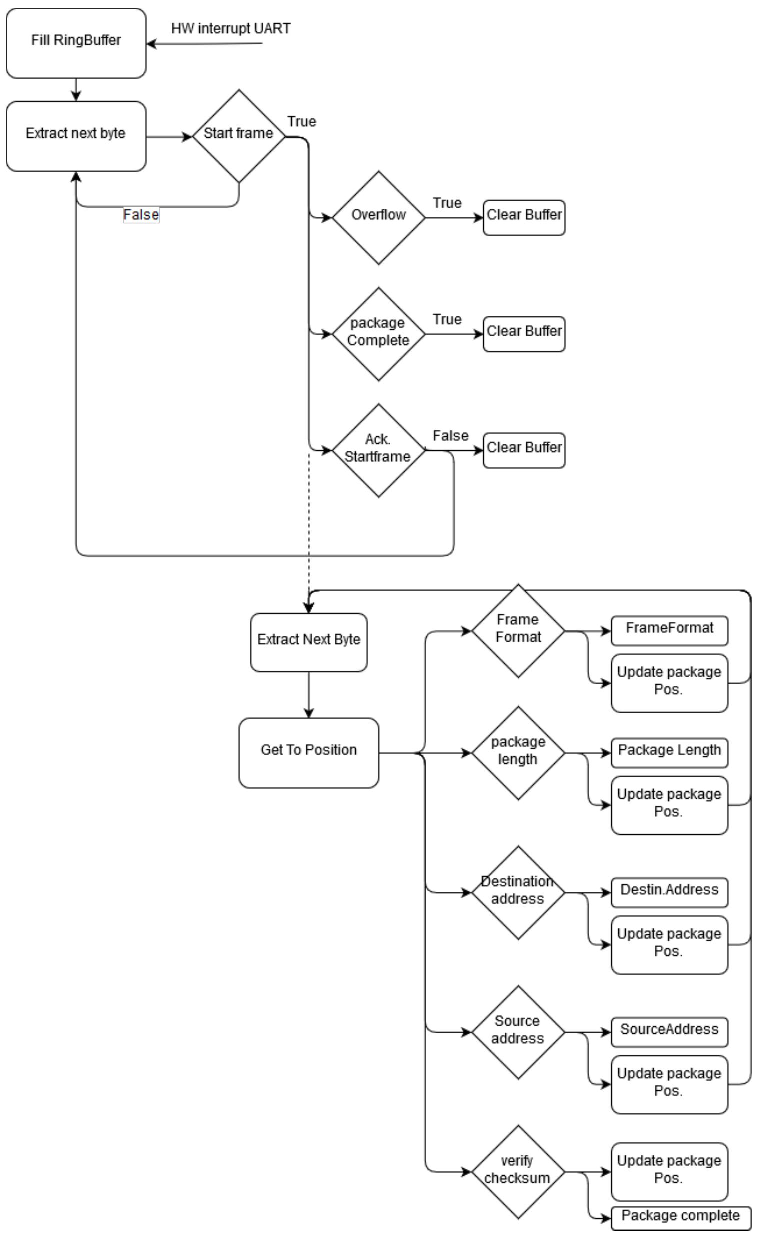


Figure 5.10: Flow chart of the DLMS interpretation procedure. Notice that this procedure will effectively produce a false output until the 7E stop flag has been detected and the message status is set as *package complete*

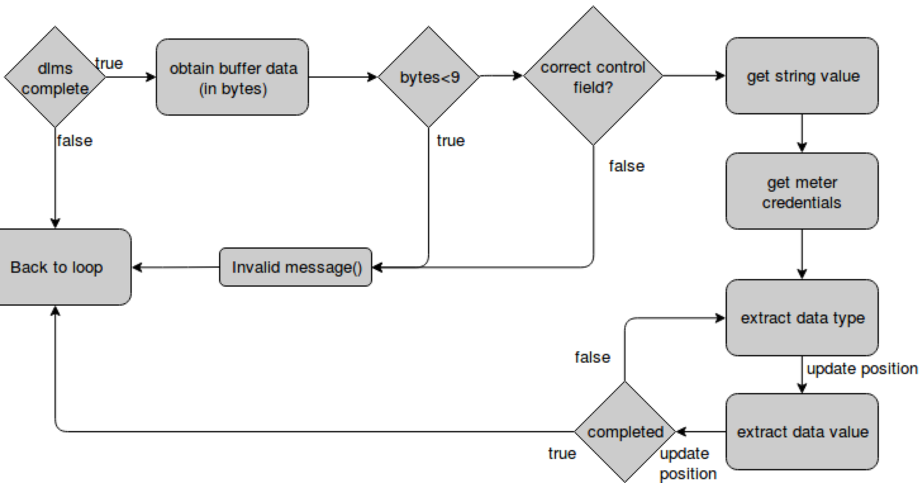


Figure 5.11: OBIS interpretation program flow chart

After the interpretation is complete all parameters extracted from the message is inserted into a JSON object. The structure is communicated to the MQTT broker as a single string. The JSON structure is depicted in figure 5.12.

5.2.3 Modem Communication

The modem communication is based on the supported programming interface provided by Digi. The driver responsible for handling the modem communication is designed by the author of a project developed in parallel to this project. The reader is advised to complement any implementation topic regarding the modem with the project report [26].

5.2.4 System Configuration

Access Point

The access point program require two libraries and two header-files: the *ESP8266WIFI*, *ESP8266WEBSERVER*, *DNSServer.h* and the custom *Configuration.h*-file. The access point functionality is only available for the ESP8266. The *accesspoint.cpp* and *accesspoint.h* are both located on the esp8266 flash memory. The access point is initiated with a setup function, responsible for enabling the debugger and check whether any configuration is present in the EEPROM of the module. The presence of a configuration file determines the state of the *runAP*-flag, a boolean value used

```
root
├── Logger ID
├── Message number
├── Timestamp
└── data
    ├── OBISlistVersion
    ├── MeterID
    ├── MeterType
    ├── ActivePowerImport
    ├── ActivePowerExport
    ├── ReactivePowerImport
    ├── ReactivePowerExport
    ├── CurrentPhaseL1
    ├── CurrentPhaseL2
    ├── CurrentPhaseL3
    ├── VoltagePhase1
    ├── VoltagePhase2
    ├── VoltagePhase3
    ├── CumulativeActivePowerImport
    ├── CumulativeActivePowerExport
    ├── CumulativeReactivePowerImport
    └── CumulativeReactivePowerExport
```

Figure 5.12: JSON structure for type 3 list AMS messages

to enable boot in AP mode. If the configuration file is detected on the EEPROM chip, a setup-function will execute an EEPROM load of the configuration file. Due to the reasons that one might want to access AP mode even though a configuration file is present a designated GPIO pin is used to trig and force the system into AP mode. This pin is defined to be GPIO 2. If the AP-flag is set, an if-condition will be entered, running the WiFi setup functions provided by the included libraries. After all, WiFi-features are active the DNS server initialization is executed. The *DNSServer.start* initializes the server with a predefined SSID name. The name will be visible when a device tries to connect to the ESP8266. After the setup is completed a loop function is executed and continuously run to handle DNS requests from the connected device.

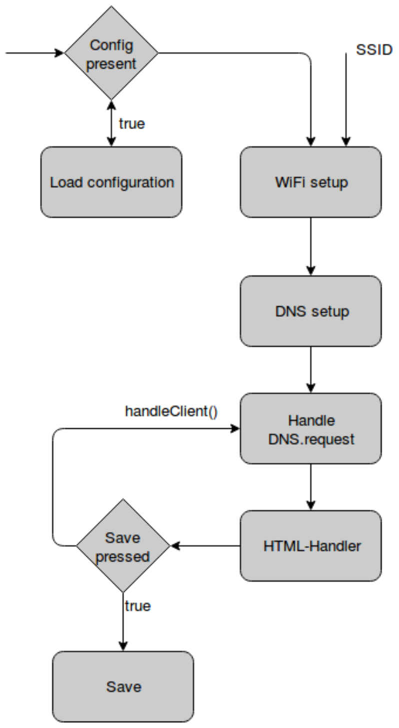


Figure 5.13: Access point program flow chart

Configuration

The configuration program require only the *EEPROM* library for Arduino. The program handles the save of credentials available in the HTML page and reading of the saved credentials. The program comprises handlers for saving, reading and storing data presented by the access point program execution. The most important feature of the configuration is the *save* and *load* feature. These two functions are responsible for saving the parameters from the HTML schema onto the EEPROM chip on the ESP8266 module and loading the parameters respectively. The configuration program is also responsible for printing the configuration to the terminal, used for debugging. Most features available in the configuration file are utility-functions necessary to make a complete and functional configuration procedure.

HTML Page

The HTML page generated by the ESP8266 during DNS AP mode is stored as a simple string in the *AccessPoint.h* file. The HTML code is written in a simple text editor and inserted into the string declaration. The solution of making one single string simplifies the HTML code handling but creates a complex string, unsuitable for effective debugging. Each insertion field in the HTML code is accessed based on a variable name, corresponding to the embedded code variables. This way of accessing variables makes the configuration and EEPROM saving procedure easy to manage.

ESP8266 Main Function

The main function of the ESP8266 is responsible for connecting to the configured WiFi network and establishing connection to an MQTT broker. The main function will also be responsible for managing the UART message reception from the LPCXpresso4367.

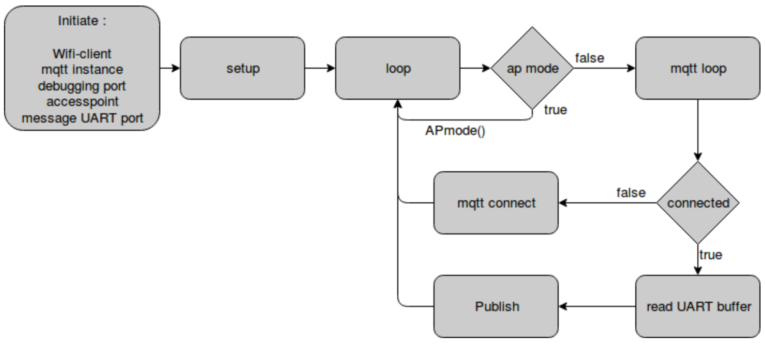


Figure 5.14: Main execution loop for the ESP8266

5.2.5 Main function

The main function of the application is implemented with only a few lines. These lines of code are responsible for initiating and executing all three tasks necessary to log and transfer AMS messages to the communication module. The first action is the hardware initialization, a small function dedicated to initiating all hardware configuration. Next is the statement of three tasks. Each of the tasks is defined in their respective library and the executed by the FreeRTOS scheduler. After each task initialization, the scheduling routines are invoked. This routine will run in a loop, effectively scheduling which task to run indefinitely. See figure 5.15.

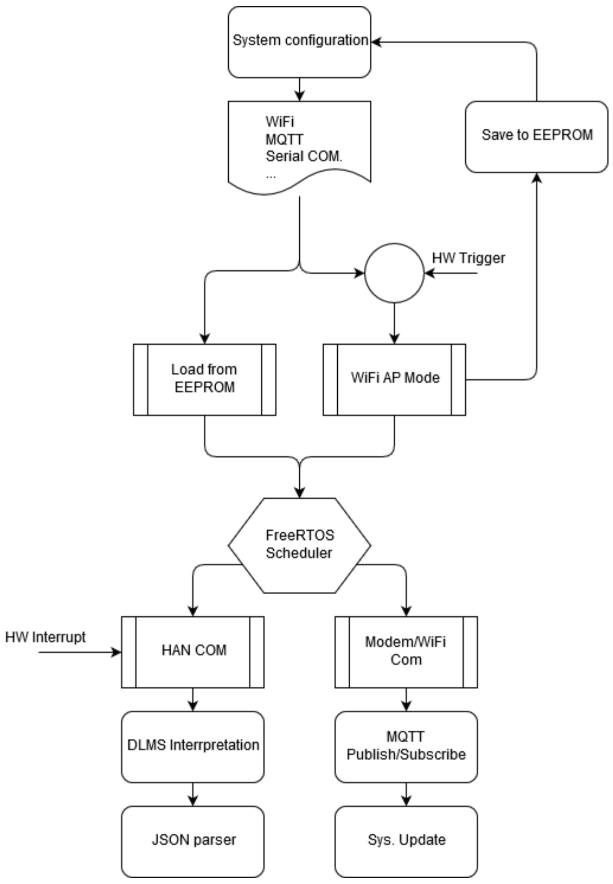


Figure 5.15: The main application execution flow with the FreeRTOS scheduler active

5.2.6 M-Bus Simulator

To achieve readable data from the simulator the software package needs to be built around existing AMS messages. These messages have been previously extracted during the previous project and have been confirmed correct. The messages are a direct translation of the M-Bus signal provided by an AMS and are therefore not restricted by the M-Bus standard message protocol. This implies that the simulator could be providing UART communication directly to the module, but for the sake of reasonable testing environments, these signals will be converted to actual M-Bus signals. The simulator software is located in a single file and executed by a round-Robin main loop. A setup function is executed in advance of the loop and configures the UART, the messages memory location, and other necessary configuration parameters. The messages are divided into three categories; short, long and long extended. Each category is subjected to a random generation of voltage values and will be renewed upon transmission. The main loop keeps track of each message sent and accounts for the rigidly defined AMS configuration. This configuration selects the message intervals as:

- Short message every two seconds.
- Long message every ten seconds.
- Long extended message every hour.

Between each transmission, the random value generator renews some of the variable parameters in the message structure and constructs a semi-unique message. Some parameters of the message will remain unaltered, but will not cripple the integrity of the system since the system should be able to handle any message, regarding the data values.

5.3 Cloud Service Utilities and Resource Implementation

5.3.1 EC2 Instance

An instance in the EC2 service is created by following most of the steps outlined by AWS EC2 user guide [11, p. 19-24]. The steps includes generation of necessary credentials and the initialization of an Ubuntu instance. All parameters belonging to the instance are set based on the *free edition* specifications. The steps are listed as:

1. Create an IAM user.
2. Set user policy.
3. Set user rules
4. Generate key pair.
5. Download the *.pem* Key-pair and convert to *.pkk* format.

6. Create instance (Ubuntu with free edition parameters).
7. Connect to instance using a desktop computer terminal and the available *.pkk*-file. Combined with SSH specifications(IPV4 and).
8. Connect and upload files to instance using WinSCP [7] on the desktop computer.

5.3.2 DynamoDB

The data table located at the DynamoDB service is constructed and initiated using the steps described by AWS. [17]. The construction of the table depended on the table name. This was arbitrary set to *AMSLoggertable*. The user-interactive webpage at AWS generated the table after the *create*-button was pressed.

5.3.3 MQTT Handler

The MQTT handler is the script responsible for subscribing to the MQTT broker from within the EC2 instance in the AWS cloud. If a message is available at the MQTT broker the MQTT handler is responsible for extracting the message and placing it into the DynamoDB data table [26]. These two features work simultaneously inside separate threads. To avoid conflicts between the two threads the MQTT handler is equipped with the *queue* structure. The utilities of the MQTT handler is divided onto two separate scripts, the *MQTT-handler.py* and the *MQTT-helper.py*. To effectively subscribe to the MQTT broker a callback-function is called in an mqtt loop. The *paho* library incorporates a predefined mqtt loop and the only thing necessary to implement is the actions to perform upon receiving a message. A designate *puToTable()*-function takes the message form the instance buffer and places it into the DynamoDB data table. To assure that the right table is used as storage the main class will also contain a predefined table name and instance.

5.4 Complete Software Design

The complete software package for the device includes the main application software, a modem communication driver, and the HAN communication driver. In addition to the device software, the M-Bus simulator stands as an auxiliary device software and the cloud management script is detained on an AWS Elastic Beanstalk distributed server.

Both drivers are defined as individual libraries and may operate independently, but do require a primary function for initializing parameters and types. The drivers can be located separately on a file system hierarchy or as part of the system's firmware.

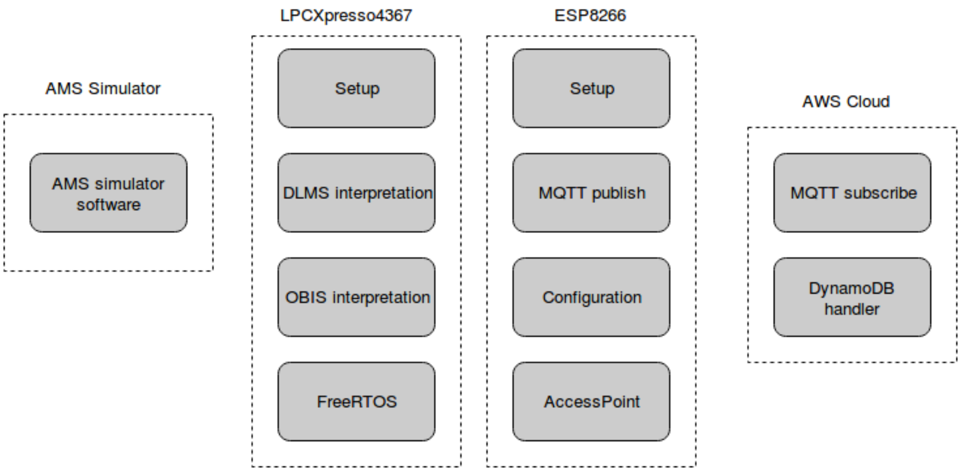


Figure 5.16: Complete software suite for the project

5.5 The Complete System Solution

Based on all preceding section the system is now complete with all modules and utilities represented. A simple graphical representation is shown in figure 5.17 with the embedded system hardware to the left and the cloud server side to the right. As the figure depicts, a couple of python scripts are necessary on the server side due to the necessity of a subscription to the MQTT broker. This may be avoided by using the IoT core service, available from AWS. The MQTT subscription program, located in the EC2 instance runs continuously and will subscribe to all messages published to the respective topic at the MQTT broker. This will allow the user to obtain a visual of the available data without significant latency.

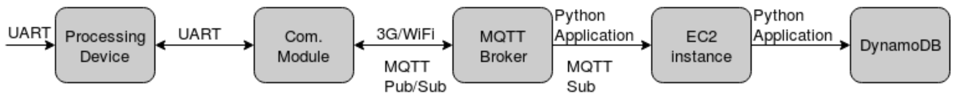


Figure 5.17: The complete system solution with hardware and cloud services

Chapter 6

Testing Procedure, Debugging and Experimentation

This chapter aims at creating the overall testing facilities incorporated into the source code or as an external procedure to assess any deviations from the specifications stated in chapter 4. This chapter lists all testing equipment and their intended functions during the testing phase. Each level of implementation will be considered individually and dedicated a single subsection. Any specification overlap is omitted to reduce chances of multiple sources of errors. The first step will account for all individual functionalities, eliminating any fundamental errors or bugs in the system source code or hardware setup. After the individual building blocks are thoroughly assessed and verified, they will be part of the simulated environment assessment. This phase will assess the system during the normal mode of operation with all modules active. The simulated environment test will also utilize the simulated AMS device to provide data to the system as if during normal operation. The chapter will refer to the specification tables 4.1,4.2 and 4.3 to test and verify whether the system operates as intended and confirms the functionality described in the project description.

6.1 Functional Testing and Verification

To properly assess the device and its functional quality the specifications are used as the base for all testing criteria. The specification table 4.1,4.2 and 4.3 will be referenced by its specification SP-xx and its correlated success-criteria TP-xx. This chapter will omit any results from these tests and merely state the progression through the functional testing of each module. Because in-field data is not available for testing, any testing of the AMS message and its content will be purely based on preemptive measurements of data acquired from a prior project.

6.1.1 Main Processing Board

To do a complete test of all system specifications a thorough verification of the main processing board is necessary to remove any errors related to hardware flaws and malfunctions. To test the board a series of small applications are constructed and uploaded to the controller, where each application has its own predicted response.

The first test investigates whether SR-2 is verified. The choice of selecting the LPCXpresso4367, with its main processor NXP4367JET100 m4 Cortex, implicitly verifies TR-2.1, TR-2.2, and TR-2.3, although USART testing will be performed separately with included software.

The first test is to assure correct upload and debugging feature using the redlink UART extension probe. This auxiliary debugging equipment allows the controller to be in debug mode without occupying the UART interrupt channel. To assure correct mode of operation a simple application program is written, featuring a led-blink response when one of the integrated buttons are pushed. This simple test is executed during debug mode. Thus a correct blinking response when the button is pressed assures that the system cooperates with interrupts during debug mode.

A second test is to assure that UART peripherals can communicate during debug mode. The test is based on a set of examples written by the openLPC group, a team constructing example projects to use on the LPCXpresso suit. The test enables UART interrupts, initiates a UART interrupt handler declares a receive ring-buffer, a transmit ring-buffer and echoes incoming UART ASCII characters to the same UART channel, using those ring-buffers. A simple serial communication terminal on a standard PC is sufficient for sending and displaying preferred characters.

A third test will assess whether the controller is capable of handling multiple UART channels with their respective interrupt handler. To test the feature of multiple interrupt handlers, a FreeRTOS task is designed for each UART. The OS scheduler is activated and starts to run after a short configuration of both UART channels. The software used in test two is reconfigured to receive and transmit on two different UART channels. The test is completed when an ASCII character is sent from the PC and echoed back using interrupts on both UART channels.

6.2 Simulated Environment Verification

This section incorporates fictional data to assess any output provided by the implemented device. The data is a predefined set of messages with public content and is therefore simple to use as verification to the system response during an actual field test. The section will not elaborate on any deviations from the specification as these will be adjusted according to any specification deviation or flaw in the system design.

6.2.1 M-Bus Simulator Circuit Test

The M-Bus simulator is tested according to the specifications of the M-Bus protocol[3]. The test is composed of a signal generator, a standard issue Arduino Nano, and the voltage amplification circuit.5.1 A power supply of *+36 V DC* is used to feed the circuit and provide the reference value for the signal. An AMS list type 1 message is manually placed in a software buffer and transmitted on the UART port to the Nano controller. The signal should propagate throughout the circuit and afflict the M-Bus output port by creating an alternating *12-13V DC* differentiated signal with a nominal voltage level of +36 V DC. The signal is assessed using an oscilloscope to assure correct output values.

6.2.2 WiFi/3G Test

The WiFi/3G test will assess whether the system can connect to a DNS server or a WiFi network. The test will be performed with predefined credentials and only account for the connection. If successful, the system should establish communication and be able to transmit and receive data via the wireless communication module.

6.2.3 System Configuration Test

The configuration feature of the system will be assessed using a simple, practical test. The system will be booted in **AP-mode** without any configuration stored in EEPROM prior to the test. The DNS server should be initialized and allow the user to connect and open the HTML configuration page. Each instance in the schema will be filled with appropriate information, like the SSID and network password for WiFi. The *save*-button will then be pressed. If the test is successful, the device should store all credentials and information in EEPROM and perform a reboot. The reboot will effectively force the system to enter **Config-mode** and initialize with the stored credentials and parameters previously defined in the HTML page.

6.2.4 MQTT Publication test

This test will verify if a JSON string is successfully published to the MQTT broker. The test will execute a publish command to a topic on the MQTT broker and subscribe to the same topic. By completing the test, the system should receive a copy of the message sent via WiFi or 3G to the MQTT broker.

6.2.5 Cloud Storing Test

This test will finalize all components of the system. The test will utilize a predefined JSON string, published to the MQTT broker. The test will include monitoring of the data table to verify the presence of the JSON string after the logger has performed a publish command to the MQTT broker. The database should catch the JSON string and store it in the DynamoDB data table without any user intervention.

6.2.6 Complete Application Functionality

A final application test is created using the M-Bus simulator circuit accompanied by a simple Arduino Nano program to generate AMS data similar to actual AMS data. The message generator transmits a message every two seconds according to [33]. The program varies between all three AMS list sizes. The main system module should then receive and process the data like any other real AMS message. The test is completed by assessing the cloud database content. To completely pass the test all modules must be able to produce the expected responses, including connection to the cloud, reception of data, message translation, transmission of processed data and provide a display of the data present on the data table in the cloud.

6.3 Test and Experimentation Setup

The complete test rig is represented in figure 6.1 and shown physically in figure 6.2. The testing rig is composed of the AMS simulation software executed on the Arduino Nano controller. The controller is mounted on the M-Bus simulator board supplied by a *36V DC* power supply. The M-Bus signal is passed through the M-Bus transceiver with galvanic insulation and into the UART port of the LPCXpresso4367. The LPCXpresso should receive the converted AMS message and interpret the DLMS/COSEM and OBIS codes part of the message structure. The interpreted message will then be converted to a JSON object and transported to the WiFi module via UART ports. The WiFi module should be configured, connected online and ready to receive the AMS messages. If received correctly the WiFi module should publish the message to the *iot-eclipse.org* MQTT server on a configured topic. A MQTT-handler located on an EC2 instance in the AWS cloud service should be running continuously, subscribe to the configured topic and retrieve the message published by the WiFi module. If all prior components work correctly the JSON string should be stored in the DynamoDB data table pre-configured in the AWS cloud service.

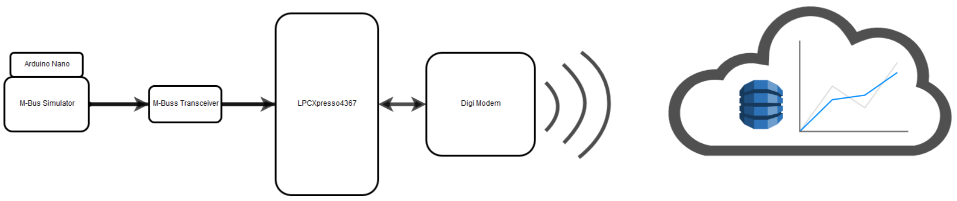


Figure 6.1: Full system test setup

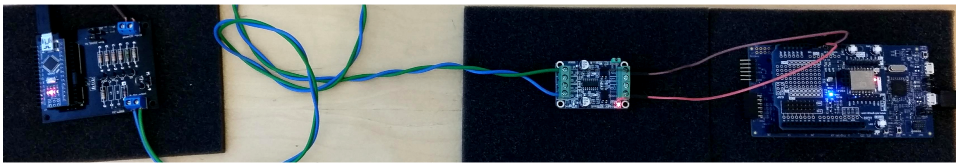


Figure 6.2: Full hardware test rig. To the right the WiFi module is seen installed on top of the LPCXpresso4367 board. The left side depicts the AMS simulator as a combination of the M-Bus simulator circuit and the Arduino Nano providing the AMS messages. The middle module is the M-Bus transceiver with optocoupled galvanic insulation.

Chapter 7

Results

This chapter aims at presenting the test results from each specification stipulated in the specification chapter 4 and its corresponding implementation 5. The results are divided into two separate categories, one for the preliminary functionality tests and one for the simulated environment tests.
The preliminary functionality tests are those tests responsible for providing a guaranteed correct mode of operation on the entire system. The simulated environment tests will display how the system reacts and responds to a simulated input and with all configurations defined. The simulated environment tests assume access to WiFi, correct cloud credentials configuration, access to the AWS cloud services and possession of a valid encryption certificate.

7.1 Hardware Assessment Results

This section lists all test results following the test phase of all hardware and software components. Each test is performed on a specific task required by the specification table 4.1. The reader is advised to complement this chapter with chapter 6 to systematically address each test-result with its corresponding testing-procedure

7.1.1 M-Bus Simulator circuit

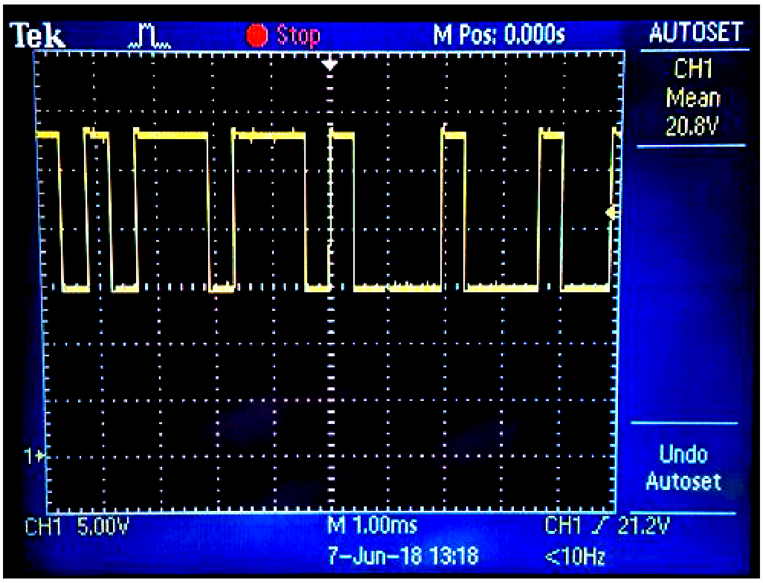


Figure 7.1: Screenshot of an oscilloscope measuring the output pins of the M-Bus simulator. Notice the 5V scale of the oscilloscope image. The voltage differentiation is clearly observable to be 13V, determined by the 13V Zener-diode.

TTL Input	Power Supply	M-Bus Output
4.7V DC	*10.0V DC*	*10.0V DC*
0.0V DC	*10.0V DC*	*0.0V DC*
4.7V DC	*28.0V DC*	*28.0V DC*
0.0V DC	*28.0V DC*	*15.1V DC*
4.7V DC	*36.0V DC*	*36.0V DC*
0.0V DC	*36.0V DC*	*23.1V DC*
4.7V DC	*39.8V DC*	*39.8V DC !!*
0.0V DC	*39.8V DC*	*39.8V DC !!*

Table 7.1: M-Bus simulator circuit test values and resulting outputs. The last two results marks the threshold for maximum voltage load before BD136 transistor overheats and shortcuts.

7.1.2 M-Bus Transceiver

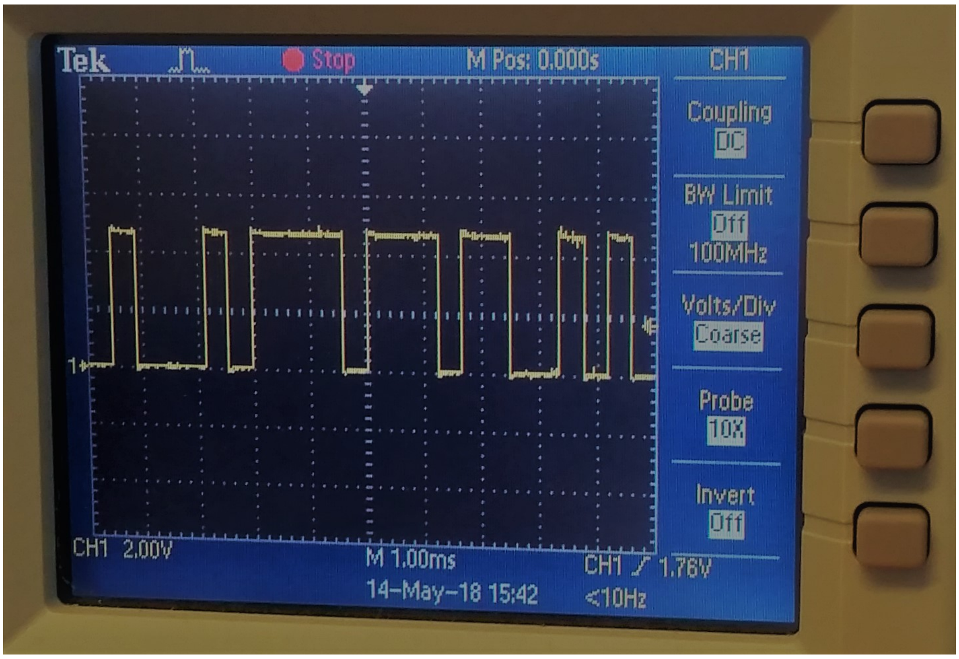
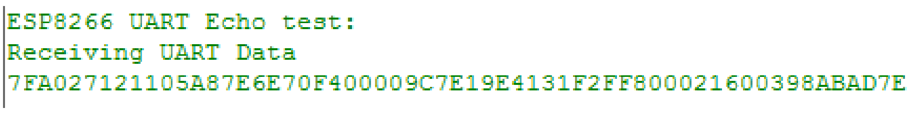


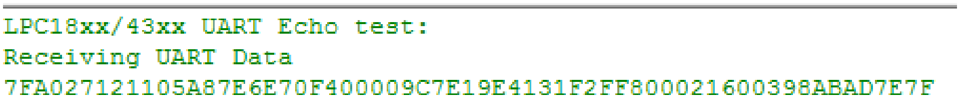
Figure 7.2: Screenshot of an oscilloscope measuring the output from the M-Bus transceiver during simulation. Notice the 2V scale of the oscilloscope image.

7.1.3 LPCXpresso4367



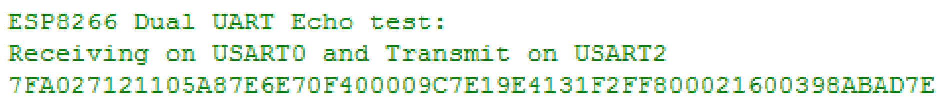
```
ESP8266 UART Echo test:
Receiving UART Data
7FA027121105A87E6E70F400009C7E19E4131F2FF800021600398ABAD7E
```

Figure 7.3: Response message from the ESP8266 during UART echo



```
LPC18xx/43xx UART Echo test:
Receiving UART Data
7FA027121105A87E6E70F400009C7E19E4131F2FF800021600398ABAD7E7F
```

Figure 7.4: Response message from the LPCXpresso4367 during UART echo



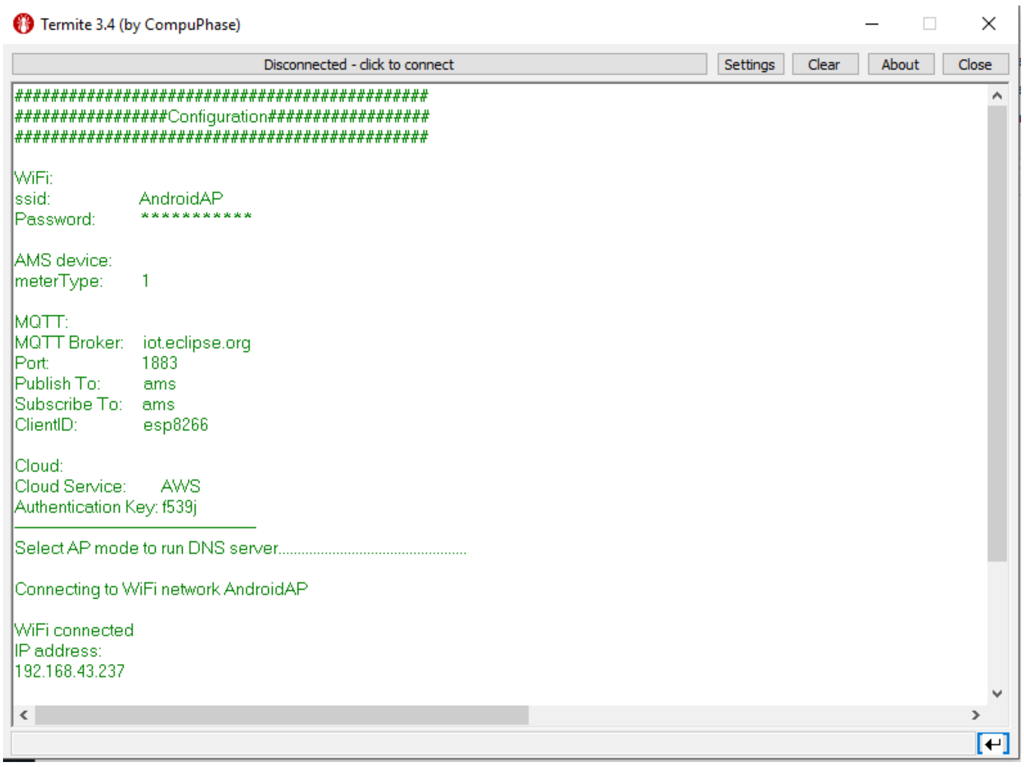
```
ESP8266 Dual UART Echo test:
Receiving on USART0 and Transmit on USART2
7FA027121105A87E6E70F400009C7E19E4131F2FF80002160039 8ABAD7E
```

Figure 7.5: Response message from LPCXpresso4367 and ESP8266 during dual UART echo

7.2 Software Assessment Results

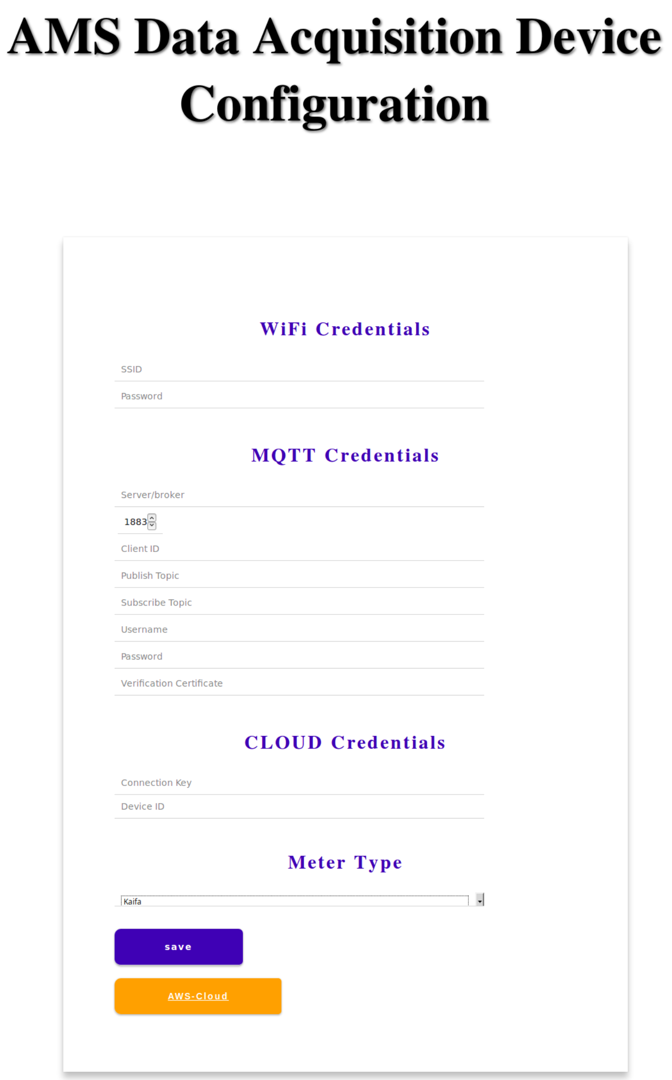
This section will display all software assessment results based on the different software feature tests described in chapter 6.

7.2.1 System Configuration Results



```
################################################
################Configuration#################
################################################

WiFi:
ssid:             AndroidAP
Password:         ***********

AMS device:
meterType:        1

MQTT:
MQTT Broker:      iot.eclipse.org
Port:             1883
Publish To:       ams
Subscribe To:     ams
ClientID:         esp8266

Cloud:
Cloud Service:       AWS
Authentication Key: f539j
____________________________
Select AP mode to run DNS server...............................................

Connecting to WiFi network AndroidAP

WiFi connected
IP address:
192.168.43.237
```

Figure 7.6: Terminal output during system configuration



AMS Data Acquisition Device Configuration

WiFi Credentials

SSID

Password

MQTT Credentials

Server/broker

1883

Client ID

Publish Topic

Subscribe Topic

Username

Password

Verification Certificate

CLOUD Credentials

Connection Key

Device ID

Meter Type

Kaifa

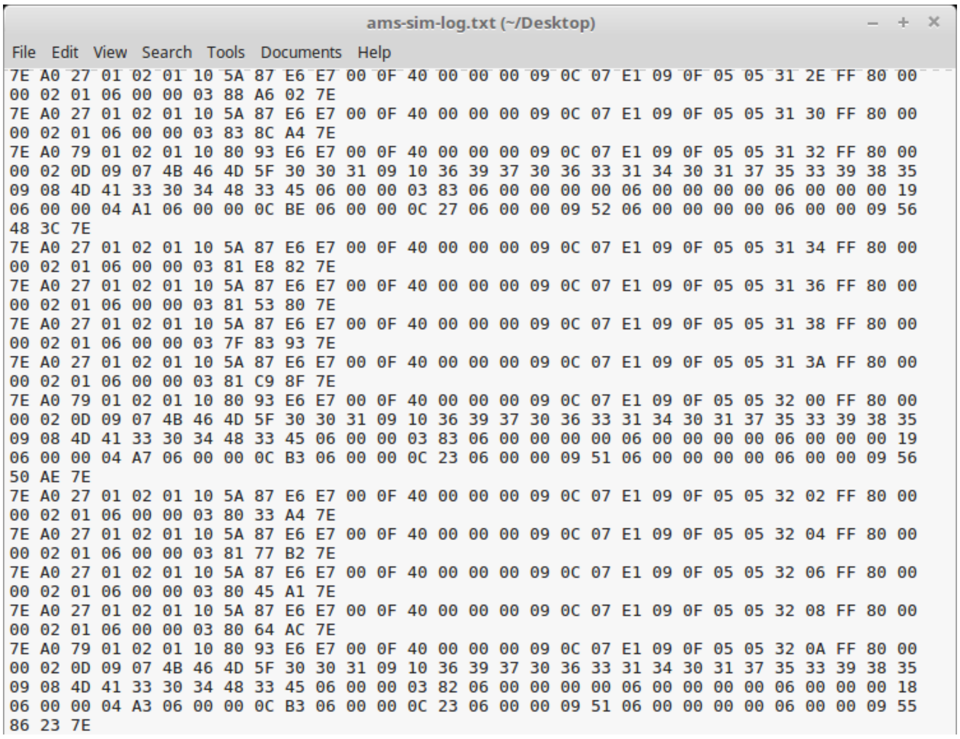
save

AWS-Cloud

Figure 7.7: Screenshot of the configuration HTML page generated by the WiFi module.

7.2.2 AMS Message Handling Results

UART Reception


ams-sim-log.txt (~/Desktop)
File Edit View Search Tools Documents Help

```
7E A0 27 01 02 01 10 5A 87 E6 E7 00 0F 40 00 00 00 09 0C 07 E1 09 0F 05 05 31 2E FF 80 00
00 02 01 06 00 00 03 88 A6 02 7E
7E A0 27 01 02 01 10 5A 87 E6 E7 00 0F 40 00 00 00 09 0C 07 E1 09 0F 05 05 31 30 FF 80 00
00 02 01 06 00 00 03 83 8C A4 7E
7E A0 79 01 02 01 10 80 93 E6 E7 00 0F 40 00 00 00 09 0C 07 E1 09 0F 05 05 31 32 FF 80 00
00 02 0D 09 07 4B 46 4D 5F 30 30 31 09 10 36 39 37 30 36 33 31 34 30 31 37 35 33 39 38 35
09 08 4D 41 33 30 34 48 33 45 06 00 00 03 83 06 00 00 00 00 06 00 00 00 00 06 00 00 00 19
06 00 00 04 A1 06 00 00 0C BE 06 00 00 0C 27 06 00 00 09 52 06 00 00 00 00 06 00 00 09 56
48 3C 7E
7E A0 27 01 02 01 10 5A 87 E6 E7 00 0F 40 00 00 00 09 0C 07 E1 09 0F 05 05 31 34 FF 80 00
00 02 01 06 00 00 03 81 E8 82 7E
7E A0 27 01 02 01 10 5A 87 E6 E7 00 0F 40 00 00 00 09 0C 07 E1 09 0F 05 05 31 36 FF 80 00
00 02 01 06 00 00 03 81 53 80 7E
7E A0 27 01 02 01 10 5A 87 E6 E7 00 0F 40 00 00 00 09 0C 07 E1 09 0F 05 05 31 38 FF 80 00
00 02 01 06 00 00 03 7F 83 93 7E
7E A0 27 01 02 01 10 5A 87 E6 E7 00 0F 40 00 00 00 09 0C 07 E1 09 0F 05 05 31 3A FF 80 00
00 02 01 06 00 00 03 81 C9 8F 7E
7E A0 79 01 02 01 10 80 93 E6 E7 00 0F 40 00 00 00 09 0C 07 E1 09 0F 05 05 32 00 FF 80 00
00 02 0D 09 07 4B 46 4D 5F 30 30 31 09 10 36 39 37 30 36 33 31 34 30 31 37 35 33 39 38 35
09 08 4D 41 33 30 34 48 33 45 06 00 00 03 83 06 00 00 00 00 06 00 00 00 00 06 00 00 00 19
06 00 00 04 A7 06 00 00 0C B3 06 00 00 0C 23 06 00 00 09 51 06 00 00 00 00 06 00 00 09 56
50 AE 7E
7E A0 27 01 02 01 10 5A 87 E6 E7 00 0F 40 00 00 00 09 0C 07 E1 09 0F 05 05 32 02 FF 80 00
00 02 01 06 00 00 03 80 33 A4 7E
7E A0 27 01 02 01 10 5A 87 E6 E7 00 0F 40 00 00 00 09 0C 07 E1 09 0F 05 05 32 04 FF 80 00
00 02 01 06 00 00 03 81 77 B2 7E
7E A0 27 01 02 01 10 5A 87 E6 E7 00 0F 40 00 00 00 09 0C 07 E1 09 0F 05 05 32 06 FF 80 00
00 02 01 06 00 00 03 80 45 A1 7E
7E A0 27 01 02 01 10 5A 87 E6 E7 00 0F 40 00 00 00 09 0C 07 E1 09 0F 05 05 32 08 FF 80 00
00 02 01 06 00 00 03 80 64 AC 7E
7E A0 79 01 02 01 10 80 93 E6 E7 00 0F 40 00 00 00 09 0C 07 E1 09 0F 05 05 32 0A FF 80 00
00 02 0D 09 07 4B 46 4D 5F 30 30 31 09 10 36 39 37 30 36 33 31 34 30 31 37 35 33 39 38 35
09 08 4D 41 33 30 34 48 33 45 06 00 00 03 82 06 00 00 00 00 06 00 00 00 00 06 00 00 00 18
06 00 00 04 A3 06 00 00 0C B3 06 00 00 0C 23 06 00 00 09 51 06 00 00 00 00 06 00 00 09 55
86 23 7E
```

Figure 7.8: Message reception on the LPCXpresso4367. The messages are displayed as read by the UART handler.

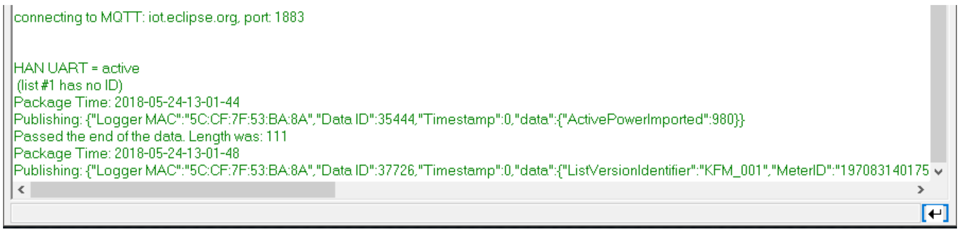
Baud Rate	Msg type 3 latency	Message Integrity
300	5.26	Overlap
1200	1.04	Valid
2400	0.52	Valid
4800	0.26	Valid
9600	0.13	Valid
18400	0.07	Bit error

Table 7.2: Message reception integrity compared to system baudrate. Valid Baud rates experienced no message errors.

DLMS/COSEM and OBIS Interpretation

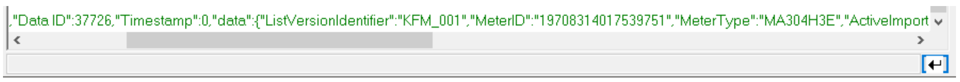
```
{
        "Logger_MAC": 5C:CF:7F:53:BA:8A,
        "DataID": 35444,
        "Timestamp": 2018-05-24-17-34-51,
        "data":
        {
                "ListVersion":KFM_001,
                "MeterID":1972681403753285,
                "MeterType":MA304H3E,
                "ActiveImportPower":0.604,
                "ActiveImportPower":0.000,
                "ReactiveImportPower":0.000,
                "ReactiveImportPower":0.156,
                "CurrentL1":1091,
                "CurrentL2":1442,
                "CurrentL3":1447,
                "VoltageL1":2402,
                "VoltageL2":0,
                "VoltageL3":2419,
                "CumulativeActiveImportEnergy":103293,
                "CumulativeActiveExportEnergy":0,
                "CumulativeReactiveImportEnergy":837,
                "CumulativeReactiveExportEnergy":21992,
        }
}
```

Figure 7.9: The generated JSON string after DLMS/COSEM and OBIS translation



```
connecting to MQTT: iot.eclipse.org, port: 1883


HAN UART = active
 (list #1 has no ID)
Package Time: 2018-05-24-13-01-44
Publishing: {"Logger MAC":"5C:CF:7F:53:BA:8A","Data ID":35444,"Timestamp":0,"data":{"ActivePowerImported":980}}
Passed the end of the data. Length was: 111
Package Time: 2018-05-24-13-01-48
Publishing: {"Logger MAC":"5C:CF:7F:53:BA:8A","Data ID":37726,"Timestamp":0,"data":{"ListVersionIdentifier":"KFM_001","MeterID":"197083140175
```

Figure 7.10: Debugging serial output after successfully translating an AMS message and initiates MQTT transfer.



```
,"Data ID":37726,"Timestamp":0,"data":{"ListVersionIdentifier":"KFM_001","MeterID":"1970831 4017539751","MeterType":"MA304H3E","ActiveImport
```

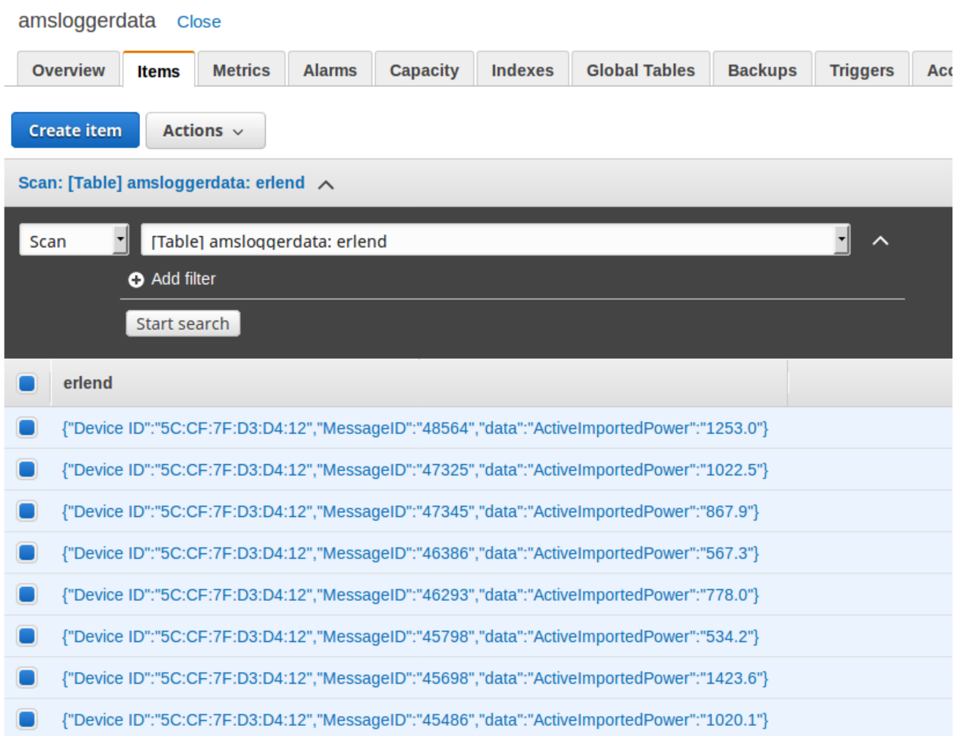
Figure 7.11: Partial image of the generated JSON string for a type 3 message

7.2.3 Wireless Communication Assessment Results

These results are Incorporated into figure 7.6 as part of the configuration of the system. Visible in the figure, the system is able to connect to the designated WiFi network and establish connection with the MQTT server.

7.2.4 Cloud Service Results

Figure 7.12 depicts the content of the DynamoDB table after receiving multiple messages from the logging device.



erlend
{"Device ID":"5C:CF:7F:D3:D4:12","MessageID":"48564","data":"ActiveImportedPower":"1253.0"}
{"Device ID":"5C:CF:7F:D3:D4:12","MessageID":"47325","data":"ActiveImportedPower":"1022.5"}
{"Device ID":"5C:CF:7F:D3:D4:12","MessageID":"47345","data":"ActiveImportedPower":"867.9"}
{"Device ID":"5C:CF:7F:D3:D4:12","MessageID":"46386","data":"ActiveImportedPower":"567.3"}
{"Device ID":"5C:CF:7F:D3:D4:12","MessageID":"46293","data":"ActiveImportedPower":"778.0"}
{"Device ID":"5C:CF:7F:D3:D4:12","MessageID":"45798","data":"ActiveImportedPower":"534.2"}
{"Device ID":"5C:CF:7F:D3:D4:12","MessageID":"45698","data":"ActiveImportedPower":"1423.6"}
{"Device ID":"5C:CF:7F:D3:D4:12","MessageID":"45486","data":"ActiveImportedPower":"1020.1"}

Figure 7.12: Snapshot of the DynamoDB table after a logging period

7.3 Simulated Environment Results

The simulated environment test did provide storage of AMS data in the cloud data table, but no graphical representation of that data were made. The plot in figure 7.13 represent the data stored in the DynamoDB table. Notice the deviation in the data point *timestamp* intervals. Each data point is collected from the *ActiveImportedPower* instance in the JSON object as this is the only instance present in all three list types of the AMS specifications. In addition to the active power the device ID is also displayed. This is the MAC address of the WiFi module responsible for sending the message and is part of the JSON object.

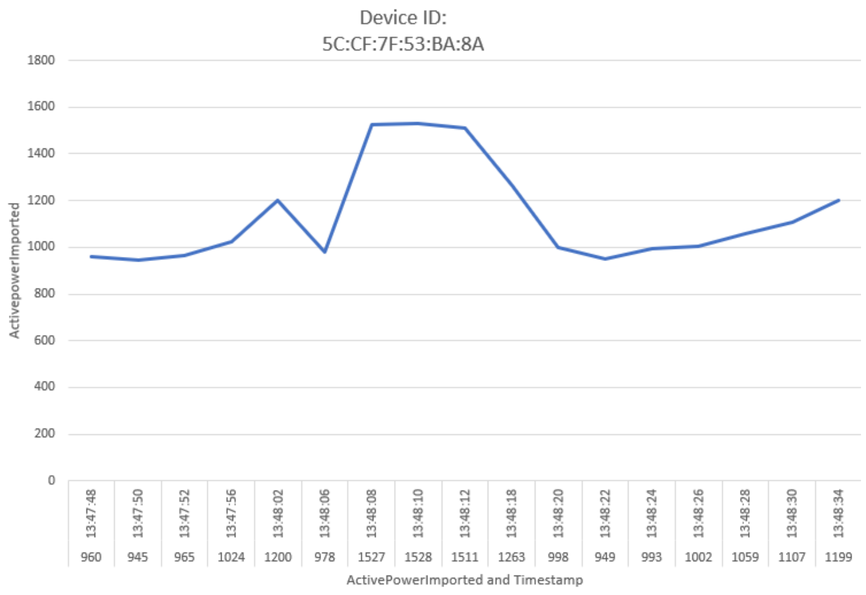


Figure 7.13: DynamoDB content plot. The plot is generated in Matlab using the derived values from the data table.

Chapter 8

Discussion

This chapter will connect the dots presented in the preceding chapters. The problem deduced from the background investigation, and its corresponding proposition of solutions presented in this report will be opposed one and another to investigate whether the test results qualify the implemented device as a feasible solution to the problem. This chapter will make use of the specification tables in chapter 4.6 and discuss how the results may affect further work on the project proposal. A thorough assessment of the system as a whole will alleviate in conclusion presented at the end of this report.

M-Bus Simulator

The testing procedure for the M-Bus simulator included the use of an oscilloscope at the output pins on the board. The result should, therefore, include an observed amplification of the TTL signal provided by the Arduino Nano. The first testing of the circuit revealed no signal on the M-Bus output pins, meanwhile a current surge was detected by observing the heat generated in the Zener diode (D1) and inability to increase the power load without increasing the supply-current. Further tests of the circuit included a systematic approach by checking each component and compare to a simulation of the circuit. Using the KiCad integrated simulation software. During the test-and-elimination approach, some deviations were detected at transistor Q2. The physical measurements did not correspond to the simulated values, as the voltage value at the collector-pin of the transistor was significantly lower than the simulated value. The tests, applied after correction of the component assembly, were successful and similar to the simulated value. The final weak spot detected in the design was the load regulating resistor R5. The value of this resistor was initially calculated to be *82Kohm*, but based on the testing results of the circuit this value was altered to relax the influence of the voltage collector at Q2. The significance of resistor R5 appeared to be clear after the change in value, as no abnormalities were detected after the change in value. The components D1 and Q4, previously generating extensive heat during a *+26V DV* power supply to the circuit, operated nominal after changing R5 to *62KOhm*. The circuit testing included increasing the power supply until one or multiple components started to malfunction. Table 7.1 show how the test was performed. The threshold was experimentally found to be approximately *39.8V DC*. The threshold value is however considered a maximum level before the diode, or the BD136 transistor fails. The reason for the current surge is unclear, even after consulting with several competent personnel. A thorough investigation in the component documentation revealed little foundation for concluding anything. All affected components should be reasonably suited for the intended application circuit, including the maximum values specified in the data-sheet for each component. The M-Bus software was based on adequate yet straightforward prerequisites and did not incorporate many features. The actual software could be designed to resemble an actual M-Bus master node. This project only required the simulator to provide predefined AMS messages, as these messages were the ones necessary for message interpretation and cloud storage.

The Test Results

During the early development of the system, a short message echo test was performed. The test would assure that the system successfully received and echoed back any input to the system module. A simple echo application was run on both the ESP8266 and the LPCXpresso4367 to assure that both modules had functioning UARTs. A short AMS message was defined as the input to the modules via PC serial terminal Termite. The response from both modules can be seen in figure 7.3 and 7.4. This result verified TR-2.2.1. For each echo, a varying baud rate was included. The range of the baud rate expanded from 300 to 115200 Baud. A baud rate of 300 - 9600 successfully echoed the message without any errors, but increasing above *9600 Baud* revealed missing bytes on the receiving terminal. Thus, the threshold for guaranteed successful echo was observed to be *9600 Baud*. The resulting limitation implies a disqualification of TR-2.1.2 but successfully verifies TR-2.1.1. Subsequently, TR-2.1 could not be considered successfully verified.

The selection of components in 4.7 was chosen to fulfill the specification. Thus, TR-2.2 and TR-2.3 were both verified successfully.

Each test committed during the testing phase was designed to disclose any short-comings in the system design. The hardware tests span from simple hardware checks to complete hardware functionality tests. Thus, when testing the software modules, one should expect every module of the system working correctly, independent of software quality. As the more complex software test revealed, such guarantee could not be made in advance of the tests. The latency between reception of an AMS message and the cache storage of the final message interpretation were significant enough to complicate transmission of the data via WiFi or 3G-modem. This issue degenerated the system not to be real-time compliant, as data would have been lost if not committed to the cloud immediately. The restriction of not having a secondary storage unit in case of dysfunctional wireless transmission degraded the system to not comply with the hard real-time demands issued in the specification chapter4.6. The loss of data should not be taken as an acceptable compromise, even though the majority of the data points were committed to the cloud successfully.

The utilization of the M-Bus transceiver with optocouplers made the transceiver independent of an external power supply as the optocouplers required a common ground power supply from the main processing board. On the M-Bus side of the transceiver, the parasitic power from the bus itself was sufficient to drive the transceiver. The transceiver was also capable of converting the M-Bus signal down to the compliant TTL voltage levels. See 7.2 subsection 7.1.2. These results successfully verify TR-1.2

Testing of system configuration was performed with a setup of credentials and specifications of meter type on an access point (AP) web page. The page should be available through a DNS connection when the system was in AP mode. To allow access to the DNS server and its pre-configured HTML page the system device was powered up and monitored through a serial connection to a random COM port on a computer desktop. A predefined debug-pin was initiated in the software suite and set to send debug-messages during run-time. The output would display if any abnormalities happened in the software suite. The debug-pin were also set to commit all configuration set by the configuration HTML page and acknowledge correct save to EEPROM. As seen in Figure 7.7 the device booted correctly in AP-mode and allowed a device with WiFi-connectivity features to connect to the DNS server and access the HTML page. The page in7.7 is displayed without any input in the schema. After all, credentials were inserted into their respective field the *save*-button was pressed and the serial terminal was monitored. The resulting messages confirmed correct save to the device and the device rebooted as expected. The procedure was performed multiple times to account for any variations in the behavior. The successful assessment of the configuration procedure approves specification TR-3.4.

System Performance

During the implementation phase, several questions appeared relevant to the selection of components. The LPCXpresso development board possesses a single core, although powerful. To incorporate a powerful chip into a system which only addresses small real-time demands could be viewed as an unnecessary dexterity, or at least ambitious. Incorporating the FreeRTOS onto the target controller verified TR-2.4 successfully. The notion of real-time systems does not immediately imply a necessity for a strong processing core, but a rather fairly adequate chip with enough integrated memory. The system was indeed tested on the much smaller and compact ESP8266 NodeMCU development board with integrated WiFi capabilities. Although the use of WiFi was not initially intended as part of the project scope, using WiFi as the mean for transmitting data greatly reduced software complexity without sacrificing functionality, except for the dependency of a locally placed router with internet access. Introducing both a 3G modem and a WiFi module successfully verified TR-3.1. When addressing the excessive use of hardware components one cannot omit the use of multiple modules separated on several boards. A complete system, incorporating the main processor, the modem, and the M-Bus transceiver, could easily have been designed to fit on a single, yet small, the board with all functionality intact. The only downside to such a solution is the increased development time and the possibility of increased costs related to the prototype board production. The costs would, however, experience a drastic decrease in the production process includes a larger quantity. This project was restricted by time and a small budget, implying that development of a fully functioning prototype board would become an overambitious approach.

After a successful echo test from both the LPCXpresso4367 and the ESP8266 device 7.47.3, a second test was performed to validate correct operation when two UART ports were used. On both devices, UART0 and UART1 were used as the UART-ports. The application was executed, with the distinction of initializing two UARTs and echo the received message on a different UART than the one used to receive the message. This approach revealed much trouble as none of the modules managed to operate as expected the first time. The ESP8266 needed some tweaking on the code, and a correction of the pin mapping before the results was satisfying. The LPCXpresso4367 occupied a ten folded amount of debugging time and investigation before the error was detected and corrected. After all, errors have handled both modules managed to utilize two UARTs at the same time. One for reception and one for transmission, although both UARTs could potentially have been doing the same check single-handed. The results were identical to that of figure 7.3.

Cloud Utilization

The solution of using AWS services imposed a strong dependency on the system's reliability. Due to the risk of unexpected downtime on the server side, the system could experience periods without cloud support. The user-data confidentiality could also be questioned, although AWS incorporates a substantial suite of security features. The management and application development related to the cloud became troublesome during the testing phase of the project. Multiple instances were launched, and many key pairs were generated. This ineffective way of managing the user credentials resulted in many hours trying to figure out why access was denied or why data was not uploaded correctly to the server. However, the final result 7.12 verified a successful upload and TR-3.3 became verified.

A shortcoming to the integration of cloud support was related to cloud management. Publishing data to the MQTT broker was considered painless and without any problems initially, but the tests revealed several downtime periods on the MQTT servers. This inconsistency of the MQTT-server availability imposed deteriorating integrity to the system. With the goal of avoiding such inconsistencies, the system should rely more on a direct connection to the cloud service, without making itself dependent on a third communication party. This would've been avoided by optimizing the modem-driver and conducting a later project with the scope of creating direct and high integrity communication channels between an embedded system and a cloud services. The broad scope of the project resulted in pooling resources with another parallel project [26]. As this project had more focus increasingly on managing the cloud server side with the mean of a 3G modem, some parts, included in this project, originated in the parallel project scope.

3G Modem VS WiFi

The specifications developed for the device were initially constructed to only account for wireless communication via a 2G/3G/4G modem, but because of multiple setbacks and difficulties related to the software testing on the modem and its correlating API, the project specification was altered to include an implementation of a WiFi module. Implementation of the WiFi module was considerably easier to accomplish compared to the modem module. Complexity in message handling between a receiver and a transmitter via a modem must take some responsibility for the complexity, as the WiFi module selected included vastly more support regarding example-code and support from the development community. Selecting a WiFi module to do the communication between the logging device and the cloud service comes with a compromise. Although implementation requirements are substantially more relaxed for a WiFi module of the type ESP8266, dependencies such as a proximity WiFi router, connected to a standard issue modem, implies reduced system robustness and less integrity. These statements are easily defended by assuming a state of the environment without access to a primary power supply source or internet access. Utilizing an external router includes a separate power supply for the router and a wired connection to a separate modem with an internet connection. This solution will be less suited for confined areas and will develop an increased total cost of the system. Thus, the solution of using WiFi as the primary communication solution is highly discouraged for any future work on this or similar projects.

AMS Message Interpretation

The implementation of the DLMS/COSEM and OBIS translation was initially a partial implementation from the former project subject. The previous implementation was written in *C++* and was customized to another hardware. The differences in hardware resulted in the necessity to rewrite most of the HAN-driver code into *C* syntax and configure the code to fit with the LPCXpresso4367 board. The reconfiguration revealed several flaws in the design, mostly related to the interpretation of the DLMS/COSEM and OBIS codes. The interpretation of the messages was fairly based on the documentation available from [3], [41] and [42], all three of them quite comprehensive. In addition to the official documentation, the supervisor provided a document with a short interpretation of the three message types, transmitted by the Kaifa AMS device. The list of the interpretations was incomplete, as twelve bytes were not accounted for in the message. By comparing the list to the official documentation, most of the missing codes were recognized and put into the interpreter application. Alas, Four bytes were still unaccounted for after the update and were not further investigated, as they had little impact on the final translation of the message. See interpretation table 3.8. This feature verifies TR-1.3. The official documentation mentioned the size of the string code and the possibility for a clock synchronization byte, but the results from translating these bytes were inconclusive and could not be explicitly explained. Several bytes were also stated as redundant or unspecified and may be removed from the message structure in the future, but will not require updated software in the application of this project as it only accounts for the predefined DLMS/COSEM values for the frame formatting.

The OBIS translation were reasonably straightforward once the complete OBIS-table were implemented and included in the HAN-driver. Problems in the previous project, related to recognizing the OBIS values, were corrected by adjusting the software driver to append the bytes belonging to each OBIS code. However, during test software testing 6, some of the messages were not recognized. This could potentially happen if the ring-buffer responsible for receiving the message were contaminated with remains from a previous message. A reset of the buffer should not be necessary due to the very nature of the ring-buffer and was not investigated due to insufficient time at the end of the project. The loss of messages implied that the system was not able to fulfill the requirement of being hard real-time compliant and should be further investigated and optimized in any future work.

System Evaluation

The system in its complete state was unable to operate under hard real-time requirements. Most of the problems related to the real-time requirements were located in the code-segment responsible for the reception of data from the AMS. The OBIS-interpreter and its joint collaborator the DLMS-interpreter produced correct translations of the HAN message most of the times. However, some messages were unsuccessfully translated during the simulated environment test. Investigations of the program execution cycle were inconclusive, although the problem may have originated in the ring-buffer and the extraction from that buffer due to the presence of trash-bits originally detected during the initial UART-tests. A future test would have investigated the effect of relaxing the conditions for receiving a valid data-package from the AMS and see how the system reacts to a partial message.

The MQTT publication experienced latency with varying significance during the simulated environment test. Because MQTT brokers might experience downtime at random intervals, they can cause the system to miss storage in the cloud. The system was designed without any redundant data-backup for acquired AMS data and would've been able to resend any lost messages if this had been part of the design and implemented. A future project could include such redundancy into the project scope, further increasing system integrity and potentially establishing a significantly more robust system.

The data present on the server side of the system was not processed beyond storage in the data table. Due to limited development time, any graphical implementation of the data table was omitted, although a complete system should include a user-friendly and potentially interactive representation of the distribution network state. The data from the simulated AMS could easily be extracted from the cloud service and plotted manually, displaying the fluctuations of the power consumption in a neighborhood. To achieve transparency on the distribution network, each AMS device should contribute to a larger map of the power consumption. Data from only one node is not necessarily wasted information, but a more comprehensive picture of the grid, generated by data from multiple meters would allow the DNO to monitor successive impacts of power consumption at a specific node.

This project, with its preliminary objective, was subjected to numerous adjustments during the implementation phase. The preliminary project description was supposed to include real data provided by an operational AMS device, installed in proximity to the system development facility. Due to an unexpected malfunction in the AMS, no such data were ever made available. Lack of in-field data induced the need for an alternative solution, and the choice of designing and manufacturing an M-Bus simulator, with appropriate software to simulate the whole AMS, were reasonably defended. The inclusion of a prototype simulation circuit required a substantial amount of development time, not only to the design itself, but also to include testing and manufacturing of the device. Adding a component development of this magnitude to the project scope restricted the amount of time available to focus on other implementation objectives. Thus, insufficient time at the final implementation of the cloud-service management reduced the development only to be a partial implementation. Missing utilities included the ability to sort the incoming JSON messages, extract data from the database and create a user-friendly representation of the database content. Although this was not considered a necessity to complete the project objective, it would've been a subtle and complementary functionality to the overall logging system.

Incorporation Of Data Acquisition Devices To The Distribution Network Infrastructure

The device designed and implemented in this project is capable of logging data from Kaifa AMS device following the HAN-standard installed on all AMS devices in Norway. An additional support for other AMS producers would require the device to possess OBIS lists specific to the desired AMS device. If each residence possessing an AMS installs a logging device such as the one described in this project the data sent to the cloud could be made available to the DNO for each respective geographical area. The data would alleviate in creating a transparent LV distribution network as the data would be a real-time display of the consumption at a specific area or neighborhood. Keeping the data in a storage facility could allow future projects to develop preemptive models for power management during challenging periods, specifically when high load and power-fluctuations influence the network. The inclusion of real-time AMS data would provide the means for establishing distributed regulation systems to the DNM and the respective network. Such integration of automation and monitoring capabilities could lift the traditional distribution network to approach the status as a smart grid, effectively regulating each section based on load profiles and real-time inputs from metering devices and their integrated loggers. To evolve the distribution network into the realm of smart grids, the data would have to be processed further in the cloud, mainly to create load profiles of a neighborhood and install load regulation equipment at the responsible transformer station.

Project Contributions

This project has made an example of how a simple yet versatile logging device could acquire data from an AMS and provide storage on a cloud server. Logging devices are no new technology, but the integration of a non-proprietary AMS message interpreter and a coherent wireless transmitter is not particularly available on the open market. To integrate a standardized logger into each node on the distribution network and collect practical power consumption information the device needs to be compliant with multiple devices. This logging device is highly customizable and scalable. These features are highly desired as many different AMS devices are installed across the distribution network. Unification of the data will evidently reduce development time and installation costs.

The construction of an M-Bus simulator circuit is particularly interesting for development projects as no actual AMS device needs to be available for testing the developed equipment. The M-Bus simulator design is compact and straight-forward to construct. Thus, the circuit could quickly be built on demand, without requiring any advanced production techniques. The simplicity of the circuit allows for private projects to take advantage of the circuit and design their custom equipment based on an M-Bus signal.

Future Improvements

Because users may want their data to be highly confidential, security measures had to be incorporated into the communication between the embedded system and the cloud service. This security was mainly incorporated into, and by, then the cloud service. Thus, little network security was incorporated into the embedded system itself. To achieve multiple layers of security, a further investigation of embedded security and confidentiality measures should be considered. Full system development should also include prototyping of a custom board with all modules Incorporated into one single unit. Further work should be initiated to design a module with the 4G modem as the sole medium for data transmission between the device and the cloud, as this would enhance the capabilities of the device. The restrictions following usage of a WiFi module instead of 4G is not merely optimal for any device intended to be part of the smart-grid infrastructure. At the server side, many future investigations could contribute to different utilities based on the data acquired from several AMS devices in a neighborhood. Below is a short list of proposed projects for any future work related to the findings of this project.

Chapter 9

Conclusion

Network transparency is a considerable problem when aiming at automating load regulation and power delivery on a power distribution network. To obtain network transparency beyond the final monitoring point, the transformer at the LV transition, a logging device with incorporated wireless communication properties could be installed at each node in the distribution network LV segment. Currently, AMS devices are being deployed throughout Norway to initialize a digitization of the distribution network and to further optimize power distribution and monitor power consumption. Thus, due to fiscal reasons an auxiliary logging device could take advantage of this power metering device and log all data provided by the AMS, using the integrated HAN port.

The project scope included development of an acquisition device with integrated features for transmitting data to a cloud service. The proposed solution was successfully capable of receiving AMS data, interpret the DLMS/COSEM message and extract the data content, categorized by their respective OBIS code. A publish routine was responsible for transmitting the data to an MQTT broker for further cloud processing. The final destination of the data was in an AWS DynamoDB data table. In addition to the original scope of the project an M-Bus simulator circuit was designed and produced to account for a malfunctioning AMS device, originally intended as the main data provider. The M-Bus simulator became a paramount component to the project scope as M-Bus data could be generated without the need for expensive additional equipment. The M-Bus simulator could also be used for other similar projects who might require an M-Bus master node for data supply. This project implemented and tested only one acquisition device but the system was designed to be as modular as possible. Utilizing several logging devices simultaneously should not create any problems. Predicting high interoperability could be justified based on the nature of the MQTT broker and the cloud service, as they are both highly scalable. Since the system could potentially handle multiple logging devices the DNO could easily use the data to obtain a load profile of a neighbourhood, as long as most nodes in the neighbourhood possesses the AMS and the implemented logging device. Any further work should emphasize interaction between several devices and the cloud. Research on handling data provided by the logger device and a response model for acting upon threshold trigger alarms could benefit the development of autonomous systems for regulating load and flow of power on the LV distribution network, especially if distributed generation is incorporated as part of the mainstream distribution network.

Chapter 10

Future Work

This chapter is comprised of a short list of proposed topics for any future work related to logging of data from AMS devices and storage in a cloud server. The list is is fairly defended in chapter 8 and the reader is advised to read chapter 8 for complementary argumentation and discussion related to the different propositions.

10.1 Propositions for Future Work

- Optimize the system for a 4G modem integration.
- Front-end application for visualizing acquired data in the cloud.
- Database management software for sorting incoming data and triggering warnings upon missing data-points or threshold violations.
- Data analysis and trend evaluation to generate a load profile of a neighbourhood.
- Construction of a preemptive model for estimation of consumption in a neighbourhood.
- Automated active load regulation to counteract fluctuations on the distribution network.
- Data-security analysis and implementation of data-encryption protocols and general verification-protocols beyond the integrated security of the cloud service provider.

Bibliography

[1] Ams + han om å gjøre sanntid måledata tilgjengelig for forbruker. https://www.ei.se/Documents/Projekt/Funktionskrav&20elm&C3&A4tare/2017/Norge.pdf.

[2] Kicad. http://kicad-pcb.org/.

[3] M-bus documentation. http://www.m-bus.com/files/MBDOC48.PDF.

[4] Mqtt wikipedia. https://en.wikipedia.org/wiki/MQTT.

[5] Smart device. https://en.wikipedia.org/wiki/Smart_device.

[6] solar cell efficiency records. http://www.pveducation.org/node/429.

[7] Winscp. https://winscp.net/eng/index.php.

[8] Avanserte måle- og styringssystem. https://lovdata.no/dokument/SF/forskrift/1999-03-11-301/KAPITTEL_4#KAPITTEL_4, 2014.

[9] Um10946, lpcxpresso4367/43s67/18s37 rev b boards. file:///C:/Users/Erlend/Downloads/UM10946-om13084_88.pdf, 2015. datasheet.

[10] International Energy Agency. Worldwide trends in energy use and efficiency, 2008.

[11] Amazon. Amazon elastic compute cloud - user guide for linux instances.

[12] Amazon. Amazon freertos, iot operating system for microcontrollers. `https://aws.amazon.com/freertos/`.

[13] Amazon. *Amazon Simple Storage Service - Developer Guide*. Amazon.

[14] Amazon. *Amazon Web Service*.

[15] Amazon. Aws identity and access management (iam). `https://aws.amazon.com/iam/`.

[16] Amazon. Aws iot core features. `https://aws.amazon.com/iot-core/features/`.

[17] Amazon. Amazon dynamodb. `https://aws.amazon.com/dynamodb/`, 2018.

[18] Amazon. What is aws elastic beanstalk? `https://docs.aws.amazon.com/elasticbeanstalk/latest/dg/Welcome.html`, 2018.

[19] Richard Barry. *Mastering The FreeRTOS^{TM} Real Time Kernel*, 2016.

[20] CIRED. *Smart Grids on the Distribution Level - Hype or Vision?*, 2013.

[21] Digi International. *DIGI XBEE CELLULAR 3G*, 2018. Datasheet.

[22] Digi International. *DIGI XBEE CELLULAR LTE CAT 1*, 2018. Datasheet.

[23] En-Mat. *EnMat Premium Data Logger*.

[24] EURELECTRIC. *Active Distribution System Management, a key tool for the smooth integration of distributed generation*, 2013.

[25] AWS FreeRTOS. *The freeRTOS reference manual*, 2016.

[26] Haakon Edøy Hanssen. Data acquisition and analysis of acquired data from geographically distributed sensors connected by 2g / 4g technology. Master's thesis, NTNU, 2018.

[27] Dent Instruments. *Eitepro XC Datasheet Rev 052215*. Dent Instruments, 2008.

[28] Texas Instruments. Meter-bus transceiver. file:///C:/Users/Erlend/Downloads/tss721a(1).pdf.

[29] Mazin H. Yassin Jessica Stromback, Cristopher Dromacque. *The potential of smart meter enabled programs to increase energy and systems efficiency: a mass pilot comparison*, 2011.

[30] Mouser. Bc-v1-ut-001. https://no.mouser.com/ProductDetail/Digi-International/XBC-V1-UT-001?qs=sGAEpiMZZMuCv89HBVkAk%2fJATD96F4I5YkJa0QjY3Po%3d, 2018.

[31] Mouser. Xbc-m5-ut-001. https://no.mouser.com/ProductDetail/Digi-International/XBC-M5-UT-001?qs=pfd5qewlna6hjdV4MSQYsQ%3d%3d, 2018.

[32] Pacific Northwest national laboratory. Home are networks and the smart grid. https://www.pnnl.gov/main/publications/external/technical_reports/PNNL-20374.pdf.

[33] NVE. Advanced metering system. https://www.nve.no/Media/5445/advanced-metering-system-ams_report_final.pdf.

[34] OSI. Osi 35.100 open systems interconnection. https://www.iso.org/ics/35.100/x/.

[35] Tim C. Green Qiang Yang, JavierA. Barria. Communication infrastructures for distributed control of power distribution networks. Master's thesis, 2011.

[36] ON semiconductor. Bc337 datasheet. https://www.onsemi.com/pub/Collateral/BC337-D.PDF, 2013.

[37] ON semiconductor. Bd136 datasheet. https://www.onsemi.com/pub/Collateral/BD136-D.PDF, 2013.

[38] Espressif Systems. Esp8266 datasheet. https://cdn-shop.adafruit.com/product-files/2471/0A-ESP8266__Datasheet__EN_v4.3.pdf, 2015.

[39] TI. *Texas Instruments*.

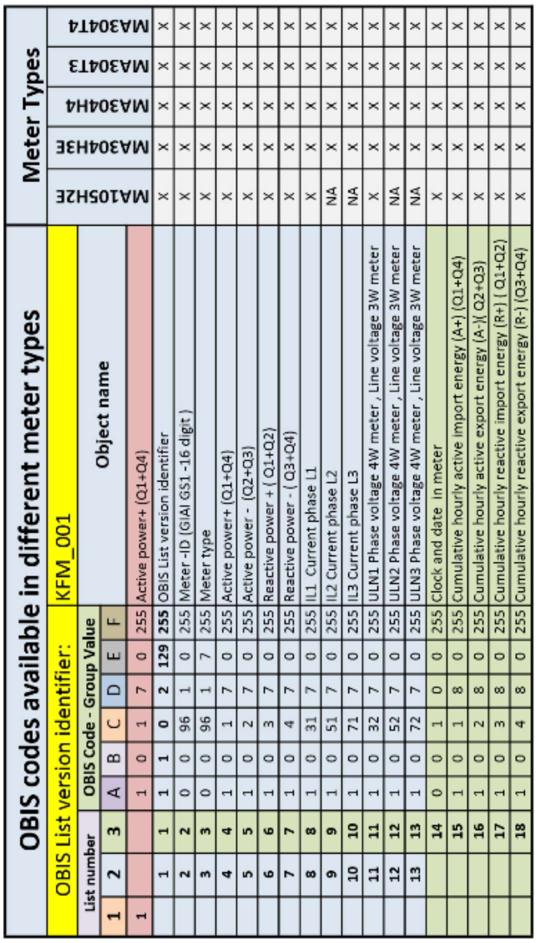
[40] Marit Schei Tundal. Using cloud services for data exchange with iot like devices. Ttk4550, Norwegian University of Science and Technology, Trondheim, December 2017.

[41] DLMS UA. *DLMS/COSEM Blue book*, 2014.

[42] DLMS UA. *DLMS/COSEM Green book*, 2014.

[43] Weatherford. Advanced metering systems. `http://weatherfordtx.gov/1632/Advanced-Metering-System-AMS`.

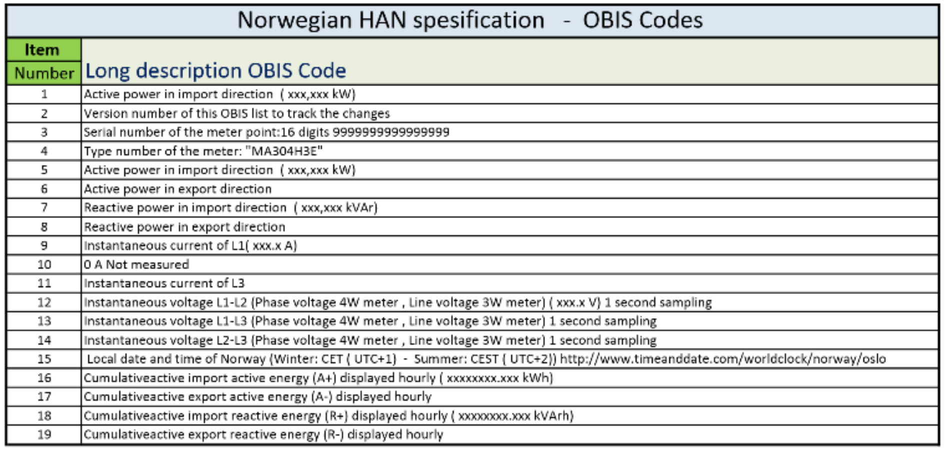
Appendix A



OBIS codes available in different meter types										Meter Types				
OBIS List version identifier:									KFM_001	MA105H2E	MA304H3E	MA304H4	MA304T3	MA304T4
List number			OBIS Code - Group Value						Object name					
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	X	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	X	X	X	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

Figure 10.1: OBIS codes for different meter types

[H]



Norwegian HAN spesification - OBIS Codes

Item Number	Long description OBIS Code
1	Active power in import direction (xxx,xxx kW)
2	Version number of this OBIS list to track the changes
3	Serial number of the meter point:16 digits 9999999999999999
4	Type number of the meter: "MA304H3E"
5	Active power in import direction (xxx,xxx kW)
6	Active power in export direction
7	Reactive power in import direction (xxx,xxx kVAr)
8	Reactive power in export direction
9	Instantaneous current of L1(xxx.x A)
10	0 A Not measured
11	Instantaneous current of L3
12	Instantaneous voltage L1-L2 (Phase voltage 4W meter , Line voltage 3W meter) (xxx.x V) 1 second sampling
13	Instantaneous voltage L1-L3 (Phase voltage 4W meter , Line voltage 3W meter) 1 second sampling
14	Instantaneous voltage L2-L3 (Phase voltage 4W meter , Line voltage 3W meter) 1 second sampling
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 (xxxxxxxx.xxx kWh)
17	Cumulativeactive export active energy (A-) displayed hourly
18	Cumulativeactive import reactive energy (R+) displayed hourly (xxxxxxxx.xxx kVArh)
19	Cumulativeactive export reactive energy (R-) displayed hourly

Figure 10.2: Descriptive list of the OBIS codes for the HAN specifications