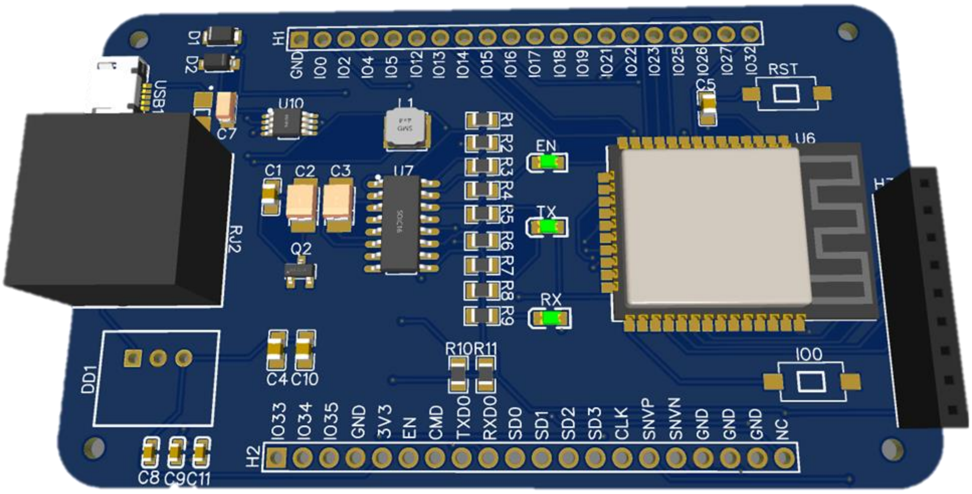
Western Norway University of Applied Sciences

BACHELOR THESIS:

BO22EB-03 - AMS HAN TO WI-FI MQTT INTERFACE WITH DISPLAY AND MOBILE APPLICATION


D1
D2
H1
GND IO0 IO2 IO4 IO5 IO12 IO13 IO14 IO15 IO16 IO17 IO18 IO19 IO21 IO22 IO23 IO25 IO26 IO27 IO32
RST
USB1
C5
U10
L1
R1 R2 R3 R4 R5 R6 R7 R8 R9
EN
TX
RX
U6
C7
C1 C2 C3
U7
RJ2
Q2
DD1
C4 C10
R10R11
IO0
H2
IO33 IO34 IO35 GND 3V3 EN CMD TXD0 RXD0 SD0 SD1 SD2 SD3 CLK SNVP SNVN GND GND GND NC
C8 C9C11

Shao Heain Tseng
Lazar Delic
Javed Qasimi

26. May. 2022

Document Control

Report title AMS HAN to Wi-Fi MQTT Interface with Display and Mobile Application	*Date/Version* 26. May. 2022/3.0
	Report number: B022EB-03
Author(s): Shao Heain Tseng Lazar Delic Javed Qasimi	*Course:* ELE350
	Number of pages including appendixes 88
Supervisor at Western Norway University of Applied Sciences Svein Haustveit	*Security classification:* Open
Comments: We, the authors, allow publishing of the report.	

Contracting entity: Western Norway University of Applied Science	*Contracting entity's reference:*
Contact(s) at contracting entity, including contact information: *Svein Haustveit* *+47 55 58 76 59* *Svein.Haustveit@hvl.no*	

Revision	**Date**	**Status**	**Performed by**
R.0.0	02.05.22	Started Project	Javed Qasimi
R.0.1	03.05.22	Fill up information Adjusted cover page	Shao Heain Tseng
R.0.2	03.05.22	Made and inserted illustrations Added contents to 1 - 3.1.2	Javed Qasimi Lazar Delic
R.0.5	04.05.22	Added contents to 3.1.2 – 3.1.6 Added illustrations	Lazar Delic
R.0.6	05.05.22	Added contents 3.1.7-3.2	Lazar Delic
R.0.7	06.05.22	Added preface, contents to 4.2 Added contents to chapter 5	Lazar Delic Shao Heain Tseng Javed Qasimi
R.0.8	08-10.05.22	Added sub-headings and contents Create illustrations	Lazar Delic Shao Hean Tseng Javed Qasimi
R.1.0	11.05.22	Added WBS diagram Added contents to remaining chapters First draft	Lazar Delic Shao Hean Tseng Javed Qasimi
R.1.1	18.05.22	Adjust chapter placement Add minor contents to 3-4	Lazar Delic Shao Hean Tseng Javed Qasimi
R.1.5	19-20.05.22	Fix project structure with PBS in mind Finish user manual first draft Adjustments to chapter 1.3-8 and appendix	Lazar Delic Shao Hean Tseng Javed Qasimi
R.2.0	22.05.22	Added hyperlinks Adjusted Appendix A-K Added GANTT diagram	Lazar Delic Shao Hean Tseng Javed Qasimi
R.3.0	23.05.22-31.05.22	Corrected grammar issues Added bullet points in chapter 4 Adjusted Chapters 1-6	Lazar Delic Shao Hean Tseng Javed Qasimi

Preface

This report is given by Western Norway University of Applied Sciences (HVL) [1], campus Bergen in the subject ELE350-1 21H Bacheloroppgave. The project was started in January and concluded in early June. The group consists of Javed Qasimi, Shao Heain Tseng, and Lazar Delic from Electronical engineering at HVL Bergen.

We would like to thank our friend and fellow student Martin Eggen for 3D-printing the case designs using his 3D-printers. We also want to thank our advisor Svein Haustveit for leading us in the right direction. Lastly, we want to thank InfoTech AS [2] for providing a professional testing area, and certain components.

1 Table of Contents

1 Introduction

1.1 Contracting entity

The Western Norway University of Applied Sciences or HVL is a Norwegian public institution of higher education. HVL was first founded in 1839 in Stord. Subsequently, the University went through many changes, the final one was being established in 2017 when five separate campuses merged into one of the largest institutions in the country. HVL offers a wide variety of academic programs at Bachelor's, Master's, and PhD levels.

1.2 Problem description

The following Bachelor thesis concerns electronics engineering, where the assignment is to use the new Advanced Measurement and Control Systems (AMS) to measure the power consumption and similar metrics and send the information to a HW solution using the Home Area Network (HAN) interface.

The task is to bring an easier way to display power usage to the general consumer, however multiple competitors already exist on the market offering a variety of solutions. The most popular solutions are created by Fjordkraft [3] and Tibber [4]. They offer an app solution that shows the current price and power usage, however, to use their services one needs to have them as their power supplier as well. This means that you cannot use their app or their services while having another supplier.

The other competition on the market are the likes of Homey [5] and Futurehome [6]. These are advanced systems that charge a high price; however, they are independent of any power supplier. They also offer a mobile app solution; though, they do not offer a display solution. For a separate display to be connected to these devices, it would have to be a DIY, or "do it yourself" project. This is not convenient for the general customer.

Keeping the competition in mind, a new, fresh, and simple solution is needed. The main issue with all the current solutions is the lack of a quick solution. They all require the app to be launched and thereafter a browsing of different menus. However, none offer a solution that can show the price in a quick glance wherever in the household. That is the main problem description to be overcome.

1.3 Main idea of solution

The contracting entity envisioned a solution where using the new AMS readers we could use microcontrollers and a Meter-Bus Transceiver (Transmitter and Receiver) to receive this data and display the power usage on a separate entity that receives the information using a cloud-based solution. The final product deviated somewhat for the main idea but withheld the same general solution of an easily accessible power metric display.

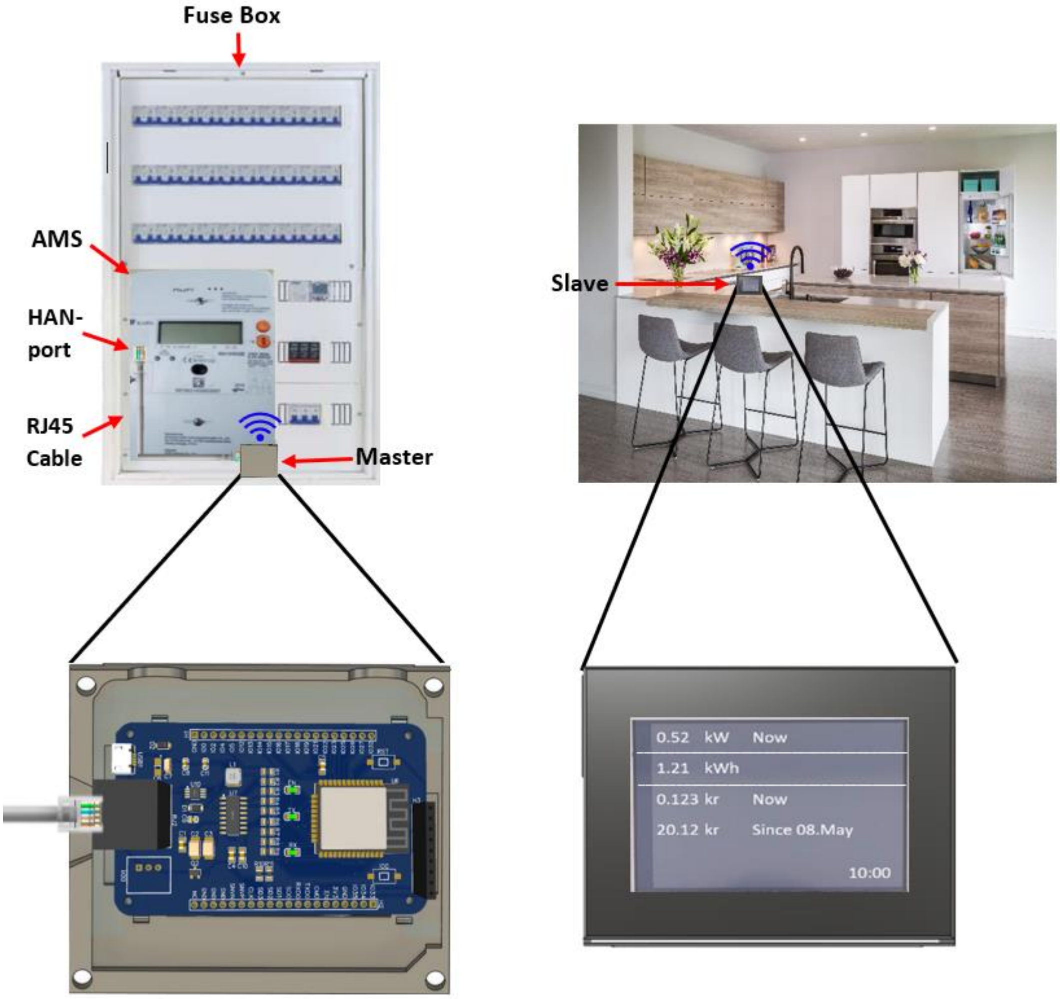


Figure 1 - General visualization of how the product works and where it is used

1.4 Failure preparations

Minimizing chance of failure is crucial in any project. Having the right preparations is key for a successful project. Therefore, following certain methodologies while working on the project proved valuable.

1.4.1 Agile

Agile was used to have an organized well-executed project. The team regularly followed Agile in the approach towards a finished product. With regular meetings and planning sessions, it was attempted to make the project as mistake-free as possible.

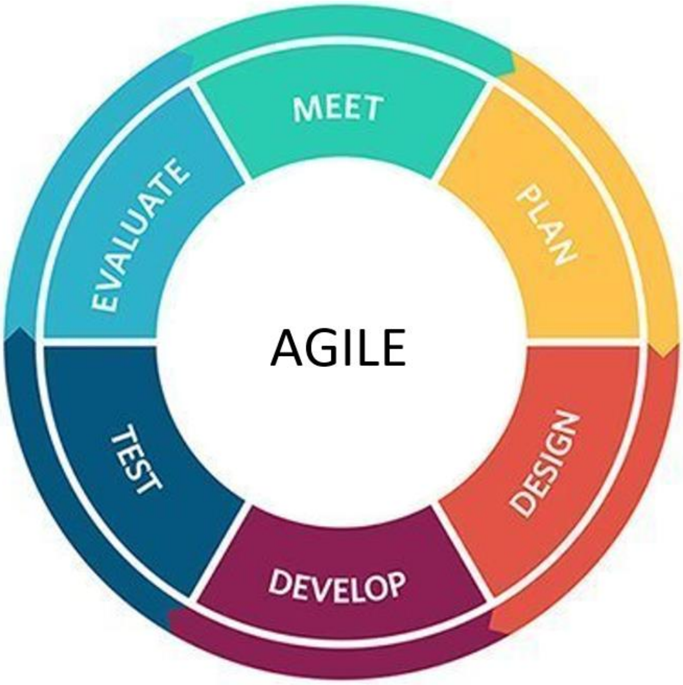


Figure 2 - Agile Diagram

1.4.2 Work Breakdown Structure

With the work breakdown structure, it is possible to have a straightforward plan, making the project more organized and predictable. The work breakdown structure lays down a foundation on what tasks need to be done.

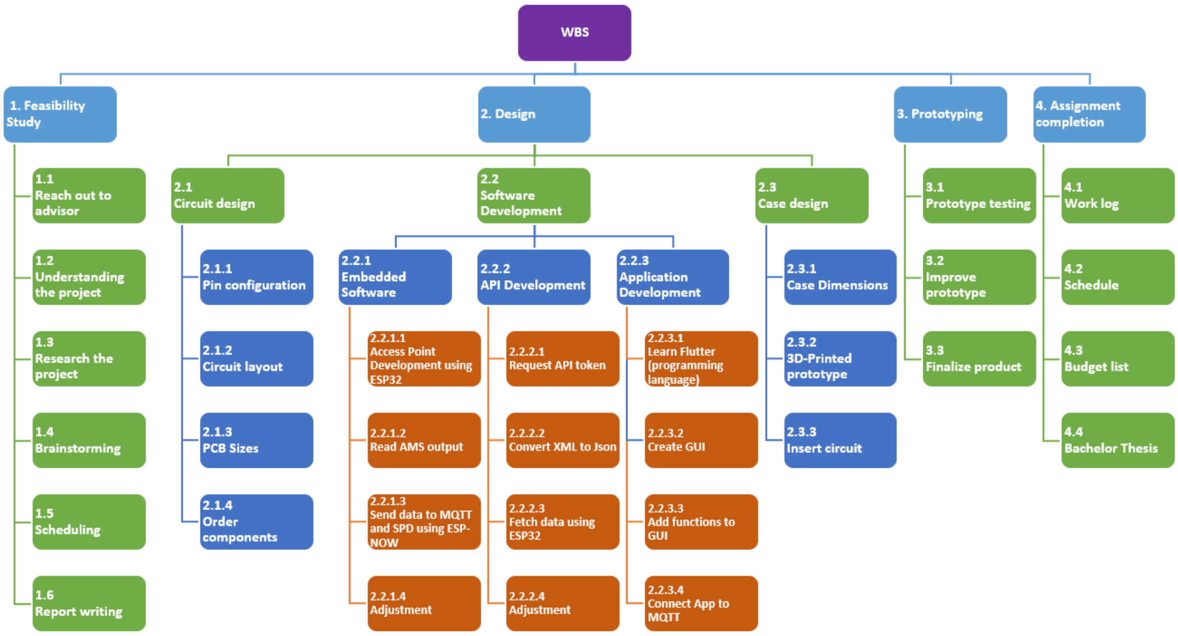


Figure 3 - Work Breakdown Structure

1.5 GANTT diagram

To visualize and plan deadlines in a structured way, using a GANTT diagram helps clarify these issues. The GANTT diagram is divided into three main parts. “Software”, “Hardware” and “Writing”. These categories consist of subcategories that summarize the work done throughout the project. Each assignment is assigned to a team member(s), and is prioritized from high (H), medium (M) to low (L). The GANTT diagram can be seen in Appendix B-A.1.

2 Specification of requirements

HVL gave us the following suggestions on how we can solve this problem. Many of these suggestions ended up being used with some slight variations to either improve performance or functionality from a user's perspective. The ideas below given by HVL were taken into consideration and worked upon to create the most well-rounded product.

Tasks

- Build Complete HW for HAN to Wi-Fi solution
- Make an embedded SW that can receive measurements from the HAN-port and send them via Wi-Fi to a Local server or to a cloud solution
- Fetch spot price call over the internet
- Build a simple display that receives measurement via Wi-Fi and shows it on the display
- Interconnect task 1 & 2 using a cloud solution server
- Receive measurements and save them on a local server or a cloud solution.
- Create a mobile app which can receive data from a local server or a cloud solution and shows the power usage in real time.

2.1 Secondary priority tasks – If there is enough time

Creating a mobile application that can receive power usage is a quality-of-life feature that is of secondary priority. However, if there is enough time to be disposed of an attempt will be made to create a simple application using "Flutter [7]". This application is to show the spot price and power usage at a given time.

2.2 Viable solutions

After considering the specification of requirements, viable solutions need to be chosen. The following subheadings consist of said solutions.

2.2.1 Spot price

The first topic of discussion is getting the spot price. This can be obtained in two ways, the first one being an Application Programming Interface (API) call made through the internet to fetch today's prices. The second way is partnering up with multiple power suppliers so that as many people as possible can use the product. The main difference between these two ways of getting the current price is the accuracy of the price. Non-partnered API calls can only get an estimated value through day-ahead prices. However, power suppliers have this information on demand, therefore it would give an exact reading of what the current price is.

What made the most sense to the group is to go with the API solution that did not require partnering up with a power supplier. While getting a precise price is advantageous, it does limit us from reaching a larger crowd of individuals. Therefore, fetching the current spot price is the better solution.

2.2.2 Hardware solutions

2.2.2.1 Single device

The problem at hand has multiple viable solutions. The first one being a single device solution. This device then must receive data from the HAN-port and thereafter send data to the cloud. To view the information sent, a mobile application also must be made. The finished product will have to have live spot price viewing as well. This is done through an API call over the internet as mentioned earlier.

2.2.2.2 Dual device

The second solution to the problem is having two devices. This makes the product unique and worth buying. As mentioned earlier, Tibber and other power suppliers already have a solution that includes a singular device connected to a mobile device. What separates this idea is, a display that can be placed anywhere in the household, showing both power usage and price. As of the time of writing, there are no other suppliers that offer a display solution on the market. The only downside to this solution is the time it takes to create two circuits which can be a tough task within a short period of time.

2.3 Choosing a solution

Since our assignment is to make a solution that can read the signals from the AMS reader, it was decided that the focus is to be on the power reading unit, later named “Smart Power Reader” referred to as SPR. The display circuit, the “Smart Power Display”, SPD, would also have a high priority; however, this would only be a priority if the first circuit is developed within the timeframe.

3 Problem analysis

At first glance, the specifications are straight forward. However, plenty of choices need to be made to fulfill the requirements set. These choices are divided into subheadings, the subheadings range from technical research to the choice of programs and components.

3.1 Advanced Measurement & Control System

Advanced Measurement & Control System (AMS) was announced in Norway in 2011 which replaced the existing analogue meters. An AMS is a modern technology for measuring and sending single or multiple phases of voltages, currents, and power consumptions to power suppliers.

By this introduction of AMS, the power companies are imposed by the Norwegian Water Resources and Energy Directorate (NVE) to make the meter data available to the customer. This is done through a RJ45 standard (ISO/IEC 8877 [8]) physical output called “Home Area Network” (HAN) interface and a standardized electrical interface Meter BUS (EN 13757-2 and EN1434-3 [9]). Customers can request to open the HAN interface from their power company.

3.1.1 AMS reader limitations

When analyzing the task at hand, a problem was discovered. There are three different AMS readers in Norway. These AMS Readers are known as the Kaifa-, Kamstrup- and Aidon AMS readers. Kaifa is mostly used in western Norway, while Kamstrup and Aidon are found elsewhere. With the group having no access to the Kamstrup and Aidon readers, the testing will therefore only be done on Kaifa readers.

3.1.2 The HAN Interface

The HAN interface is the sole reason for this project being possible. The HAN-port delivers various data ranging from momentary power usage to power usage accumulated within the last hour. Underneath is an example of where the HAN-port can be found on any new digital electricity readers.

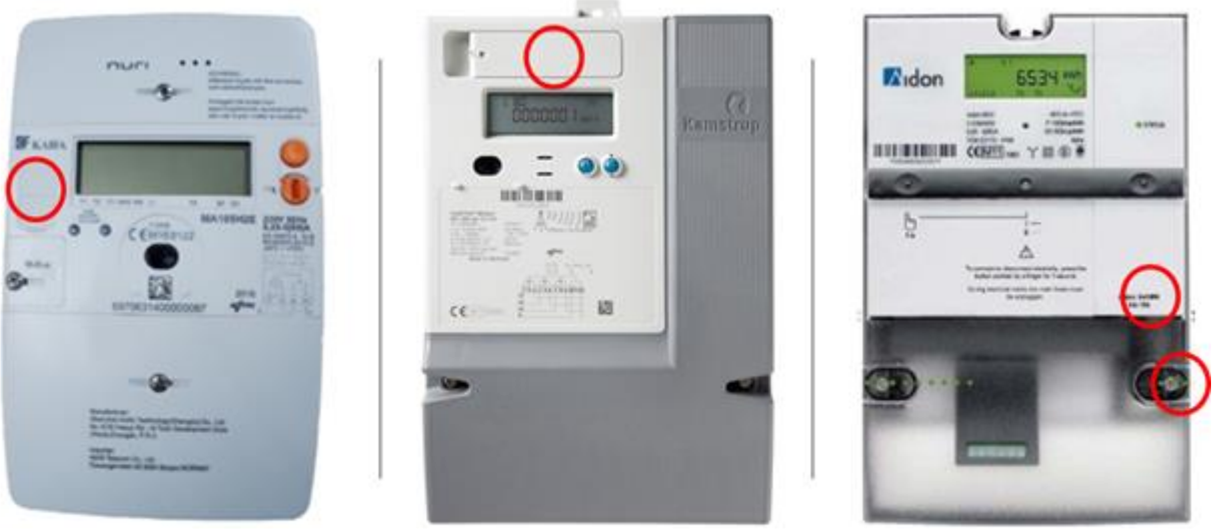
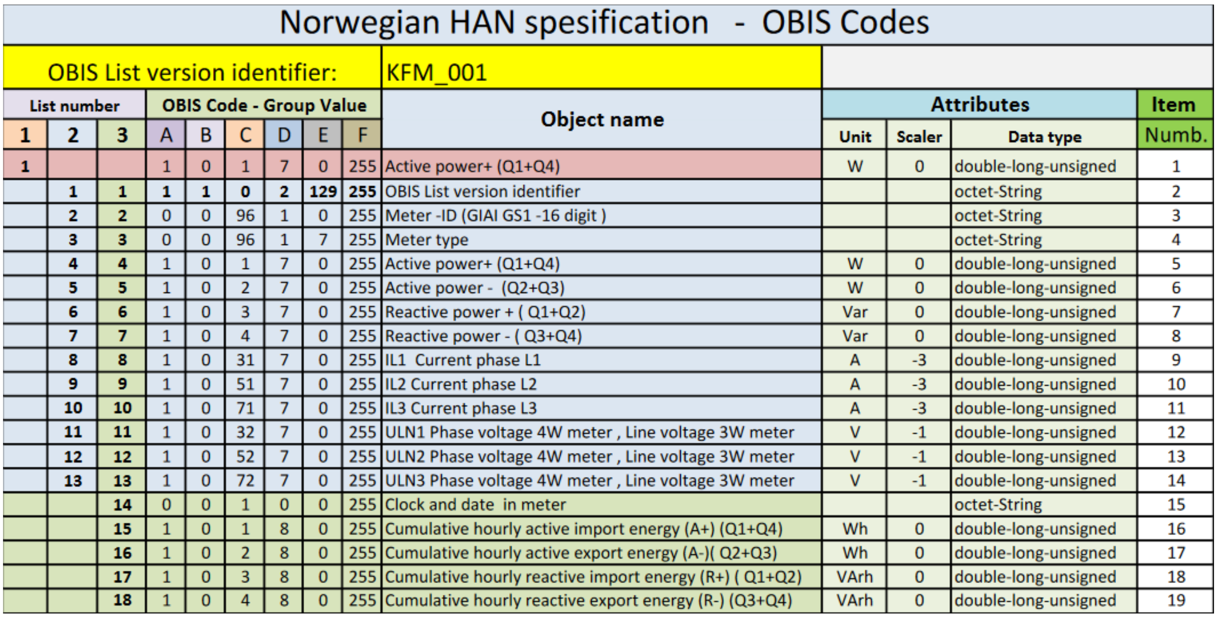


Figure 4 HAN-port location on Kaifa, Kamstrup, and Aidon AMS readers

The HAN interface delivers three different packets. The first packet gets sent every two seconds. The most important data this packet delivers is active power usage given in Watts [W]. The second packet comes every ten seconds and sends a more substantial amount of data than the smaller packet, as seen in Figure 5. Packet number three sends a packet every hour, which includes the most important data that is the power usage accumulated within the last hour given in Watt-hour [Wh]. The HAN interface offers access to plenty of other information; however, the general consumer is most intrigued by the power usage, and power usage per hour. Therefore, only the two mentioned data will be fetched.

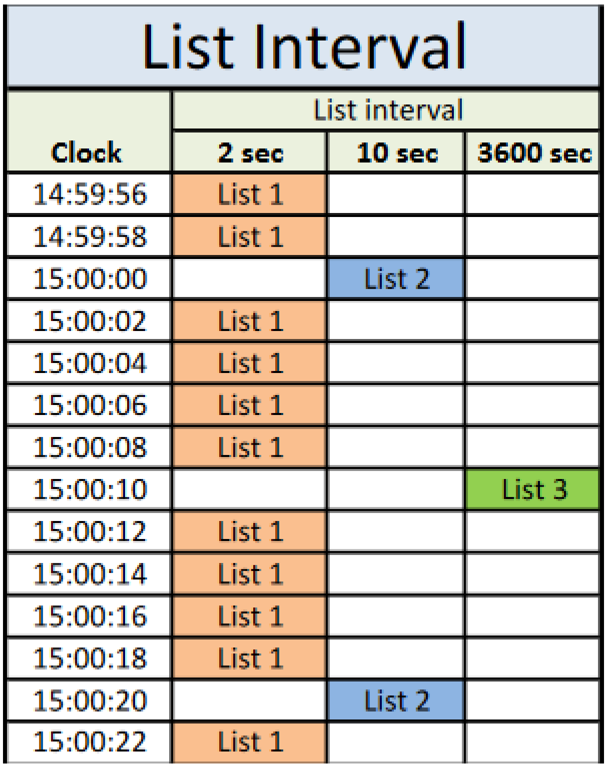


Norwegian HAN spesification - OBIS Codes

OBIS List version identifier: KFM_001

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

Figure 5 - Overview on all data sent via the HAN interface



List Interval

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

Figure 6-List showing when the different packets arrive

The current idea is to only use the data sent for power usage and power usage per hour. However, it is completely possible to integrate other data in the future. An example is the date and time being delivered every two seconds, this can be processed and shown on the screen. The same applies for most of the data shown in Figure 6 as well as in Appendix K.

3.1.2.1 M-Bus Protocol (Meter-Bus Protocol)

The M-Bus Protocol is a protocol that allows computers to read energy consumption data from power meters, heat meters, gas meters, water meters and various sensors and actuators from different manufacturers. M-Bus support remote powering of slaves and works in a Half Duplex fashion. The wire named M-Bus+ is kept at constant 36V to supply the slaves with power. The other wire M-Bus- alternates between 0 and 12V, giving either 36V or 24V in difference between M-Bus+ and M-Bus-. The difference between M-Bus+ and M-Bus- determines if it is a logical one or zero being transmitted. 36V between them corresponds to a logical "one" and 24V to a logical "zero". Based on the M-Bus description documentation, the protocol used between a master and slaves is an asynchronous serial bit transmission. This is visualized on the Figure 8.

To get the transmission to work, the first requirement is that the transmission between two telegrams must have no delay. The second requirement is that the eleventh bit, which is the "stop" bit, must be a logical “one”, while the "start" bit must be a logical “zero” to start transmission. As seen in Figure 8, the bits are sent in an ascending order, least significant bit first (LSB).

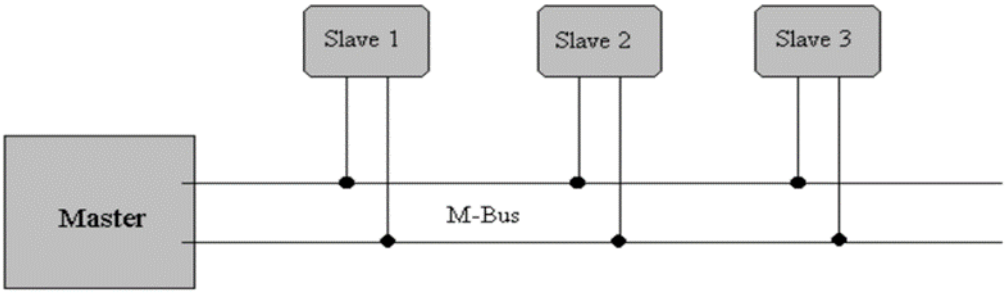


Figure 7 - M-Bus master/slave connection

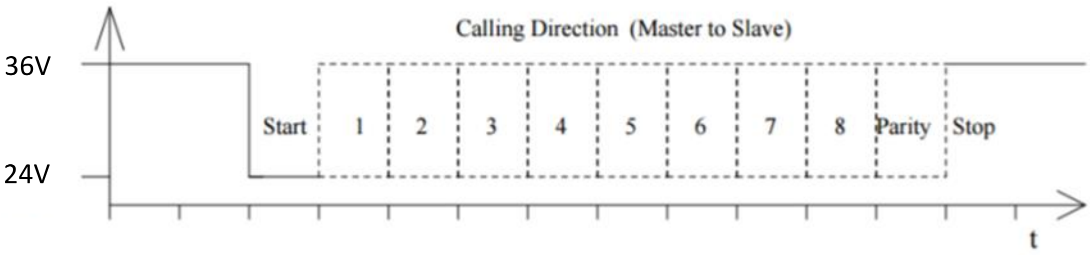


Figure 8 - M-Bus transmission packet

3.1.2.2 Universal Asynchronous Receiver/Transmitter (UART) Protocol

Universal Asynchronous Transmitter Receiver (UART) [10] is a hardware peripheral protocol that can be found inside microcontrollers. UART converts the incoming- and outgoing data into the serial binary stream. 8-bit serial data received from the peripheral device is converted into the parallel form using serial to parallel conversion. Likewise, the parallel data received from a Central Processing Unit (CPU) is converted using serial to parallel conversion. The asynchronous serial communication data and speed is also configurable.

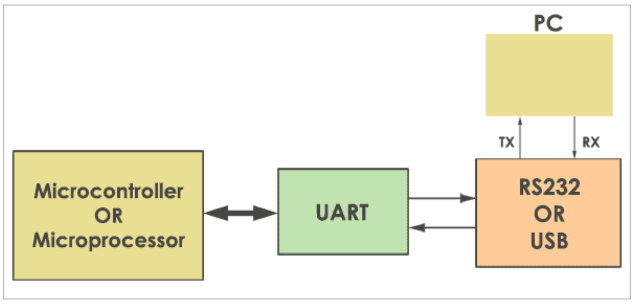


Figure 9 - UART block diagram

Serial Peripheral Interface (SPI) and Universal Serial Bus (USB) are often used for fast communication, while UART is most often used at lower data transfer speeds. UART is a cheap communication device with a transmitter/receiver. One wire is required from transmission of data while another wire is needed for receiving data. Using an RS232-TTL converter or USB-TTL converter one can communicate with a Personal Computer (PC). The common thing between RS232 and UART is that they are both asynchronous, meaning that they don't require a clock to communicate. UART consists of one start bit, 1 or 2 stop bits and a parity bit for serial data transfer.

3.2 Hardware development

When designing a hardware circuit, it is important to be able to verify the circuit through simulations and circuit analysis. To achieve this, a Printed Circuit Board (PCB) design tool that offers a large variety of components is essential. Be as it may, there are no design tools that have all the necessary parts for this project and simultaneously support live simulation on said parts. Therefore, we must settle for programs that allow us to gather all the components into one singular program.

As a result, it is decided to use EasyEDA [11] as the main PCB design tool. This is simply because there is no other tool that included all the components we laid forth. Finding a single design tool that has all the components makes the order and manufacturing process more organized.

3.2.1 Main components

Based on the specifications given, Wi-Fi access on the circuit is a crucial factor for both cloud access and for fetching spot price using the internet. Therefore, it was decided that the ESP32-WROOM-32D [12] microcontroller is a solid fit for this assignment. Using this microcontroller we conquer specifications, namely, communication with cloud solutions and -APIs. The ESP32 was sub-consequently also chosen because of the group's knowledge of the Arduino IDE and C++ programming. This effectively leads to a smoother starting phase.

In addition to the ESP32 we also needed other components to achieve the goals set. For the chosen microcontroller to understand the signals delivered from the AMS reader, a Meter Bus (M-BUS) Transceiver is needed. There were multiple acceptable solutions on the market. The first being NCN5150 [13], and the latter being TSS721A [14]. Both allow microcontrollers to understand the signals sent from the AMS reader. The signals sent are M-BUS signals that need to be "translated" to Transistor-Transistor Logic (TTL) signals.

3.2.1.1 Supporting components

When creating a complete circuit there are plenty of components involved. The main components have already been introduced. The ESP32, and TSS721A/NCN5150 (M-BUS to TTL converters). However, there are some lesser components that also have an important job. R-783.3-1.0 [15] is a 5-volt to 3.3-volt converter. This is something that is bound to the ESP32 microcontroller and is needed for the ILI9341 [16] display to obtain enough power using an external power supply. In addition to the microcontroller, an attempt is made to standardize 3.3V in all parts throughout the circuit. This thought process is done so that the circuit must have as little converters as possible.

The LT1936 [17] is the second buck-converter that is to be implemented into the circuit. This converter should make it possible to use only the HAN-port as a power supply to the SPR, meaning that the one cable connection should be fulfilled.

Lastly it is important to choose resistors, capacitors, transistors, and coils carefully as these are essential for stabilization and amplification of signals.

3.3 Software development

When doing software development, it is important to get familiar with the task and the different protocols used in the project. To fulfill the specification of requirements, it requires deep analysis of the software specification.

3.3.1 Getting spot price using API

Fetching the spot price in each power region is a considerable task. The first part is finding an API that allows for commercial use while using their services. As there were no Norwegian distributors that allowed for such use, we ended up using a European platform, Transparency European Network of Transmission System Operators for Electricity (ENTSO-E) [18]. The advantages here are the amount of API calls allowed per minute. The European platform allows a total of up to 400 calls. Seeing as the API needs to be called only once per day, this would allow for up to 2.3 million calls with just four separate tokens.

3.3.2 Azure Functions

The main reason for choosing azure functions is because of its ability to deliver data to any device using API. This means that any device that can make HTTP-call can receive the data from the cloud-based solution. Additionally, the main reason for having to use Azure Function is because of the lack of available open-source APIs for day ahead prices. There are no free solutions on the market that offer a straightforward solution. Most solutions are either paid solutions, or solutions that need to be processed further to access the data needed. Therefore, making code that only sends relevant data like spot price at any hour during the day is necessary. This would take away repeating code of processing Extensible Markup Language (XML) and other formats into raw data, while also offering a solution that only gives the data a person needs.

Another advantage to Azure Functions is that the cloud service costs scale with on-demand calls. This means that the cloud service does not accumulate high prices unless there is high demand. In addition to relative price costs, using Azure Functions makes it possible to publish the code to the cloud service using Visual Studio. This means that if any bugs were to occur, it is easily updatable and fixable using Azure. Additionally, with a history in C# programming, it made the decision easier and more straightforward. The data received from Azure Functions can as a result be received as a regular function within most programming languages.

3.3.2.1 Using Norges Bank API to convert Transparency data

When getting the spot price through Transparency ENTSO-E, the data firstly comes in as EUR/MWh. This data needs to be converted to NOK/kWh to meet the Norwegian standard. This conversion can be done through visual studio using C#. However, the current spot price depends solely on the power region a person lives in, and the current exchange rate between NOK and EUR. The exchange rates are publicly available on Norges Bank [19] API and the values are delivered using the JSON (JavaScript Object Notation) format.

3.3.2.2 Spot price data processing

Norway is divided into 5 individual power zones shown in Figure 10. The general citizen is not familiar with their power zone. That being the case, a simple solution where the power zone is not used would be optimal.

The first solution example is location tracking using GPS; however, this would prove costly and there might be privacy concerns as a result. The idea that ended up sticking because of its simplicity was a postal code solution. As seen in Figure 11 below, Norway is divided into 9 different postal codes. These postal codes create a pattern that can be seen within the power zones as well. For example, postal codes nine and eight equate to power zone four. Using this trend, one can estimate the power zone using postal codes. This also solves the privacy issue, as through postal codes, only a general area is sent, meaning that the precise location is not used.

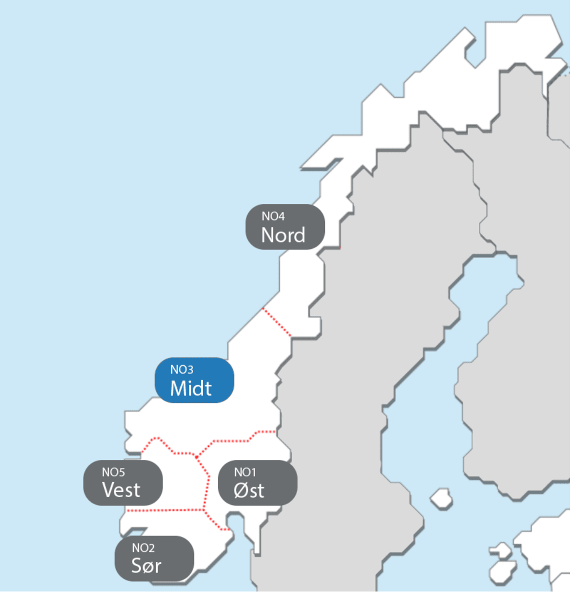


Figure 10 - Power zone distribution in Norway

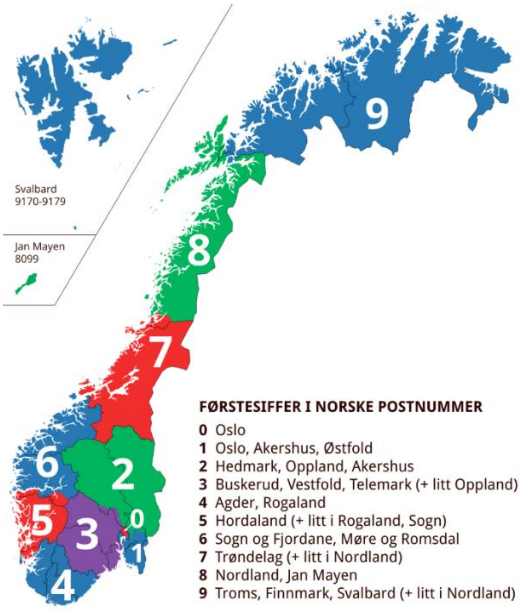

Svalbard
9170-9179
Jan Mayen
8099
9
8
7
6
2
5
3
0
1
4
FØRSTESIFFER I NORSKE POSTNUMMER
0 Oslo
1 Oslo, Akershus, Østfold
2 Hedmark, Oppland, Akershus
3 Buskerud, Vestfold, Telemark (+ litt Oppland)
4 Agder, Rogaland
5 Hordaland (+ litt i Rogaland, Sogn)
6 Sogn og Fjordane, Møre og Romsdal
7 Trøndelag (+ litt i Nordland)
8 Nordland, Jan Mayen
9 Troms, Finnmark, Svalbard (+ litt i Nordland)

Figure 11 - Postal code distribution in Norway

3.3.3 Software protocols

3.3.3.1 MQTT interconnection

MQ Telemetry Transport (MQTT) is a communication protocol based on a publish and subscribe system. The illustration below explains what this means.

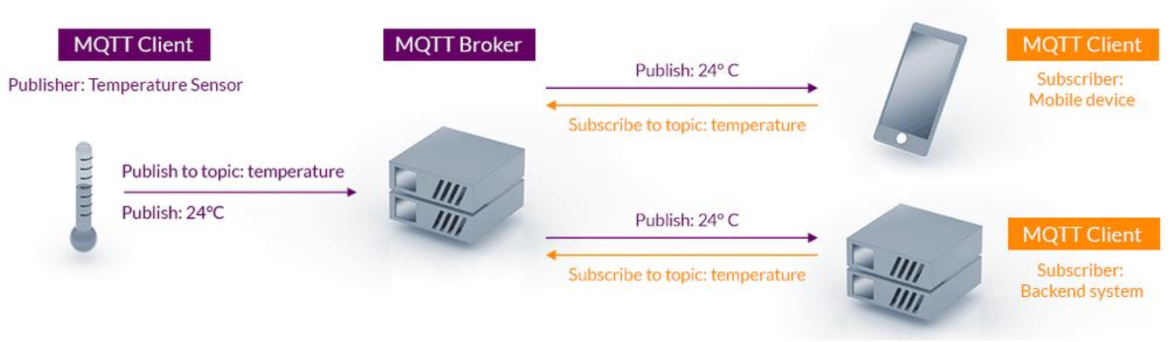

MQTT Client
Publisher: Temperature Sensor
MQTT Broker
Publish to topic: temperature
Publish: 24°C
Publish: 24° C
Subscribe to topic: temperature
MQTT Client
Subscriber: Mobile device
Publish: 24° C
Subscribe to topic: temperature
MQTT Client
Subscriber: Backend system

Figure 12 - MQTT publish/subscribe example

As seen above, it starts with a client that sends data to a broker. In this case that would be power usage, thereafter this data can then be accessed by MQTT clients that subscribe to the "topic", the topic being "power usage". After subscribing to a topic, data like power usage and power usage per hour can then be processed as an integer or a floating point. This use case is intended for the mobile application to receive the data and show it to the user.

As mentioned, it is intended to use this protocol to send and receive power usage through the internet. The broker that has been decided is Ubidots [20]. Ubidots has a good solution for students and enterprises such as limitless free usage, while also having a large database that stores data from up to two years. The amount of data stored per day can be increased with a subscription fee per month. Therefore, it is a good solution if this were to be an expanded business.

Furthermore, Ubidots offers API Tokens as verification, this makes the communication safe, and password protected. What is more, through this token, it is possible to fetch all data and send it to a mobile application using an API call. This then opens for the possibility to view power usage in the home from anywhere in the world. If the mobile device has internet connection and the correct token, it is possible to view the power usage and spot price from anywhere. This is how API calls work and was one of the main reasons for using this MQTT broker.

3.3.3.2 The ESP-NOW protocol

As for communication between the SPR and the SPD it was initially planned to use MQTT for communication. However, since the ESP32 supports ESP-NOW, it is determined that to reduce traffic to the MQTT server, ESP-NOW will be the main communication method between the two devices.

ESP-NOW is a protocol defined by Espressif [21] which sends data using 802.11 packets without the need of being on the same Wireless Local Area Network (WLAN) network. ESP-NOW allows for multiple devices to be connected without using Wi-Fi. For the connection to be established, the sender (SPR) needs to get the Media Access Control (MAC) Address of the receiver (SPD). With the protocol being low power, this means that the circuit itself will use less power overall, making it more possible to fulfill the one cable connection without power delivery issues. Additionally, ESP-NOW sends packets of 250 Bytes, making it possible to send large packs of data every second. This packet size is more than enough for this assignment.

The ESP-NOW protocol is to be used to send and receive loads of data. Among this data is, current spot price throughout a given day, power usage and power usage per hour. The data is sent using structs in C/C++ and received as raw data. In this case the power usage received will be given as a floating point, while the spot price is delivered as a floating-point array containing the price for every hour during the day.

3.3.3.3 Protocols working together

The initial thought is to have the two devices, SPR and SPD working together as master and slave respectively through ESP-NOW. The SPR is going to send power usage to MQTT using Wi-Fi and the SPD using ESP-NOW. For a connection to be established with ESP-NOW, the MAC Address of the receiver (SPD) needs to be registered to the sender (SPR). Doing the data transfer in this manner makes it so that the SPD never has to connect to MQTT, saving power and making the code faster. After the data gets delivered to MQTT, it is safe to get accessed using a mobile application.

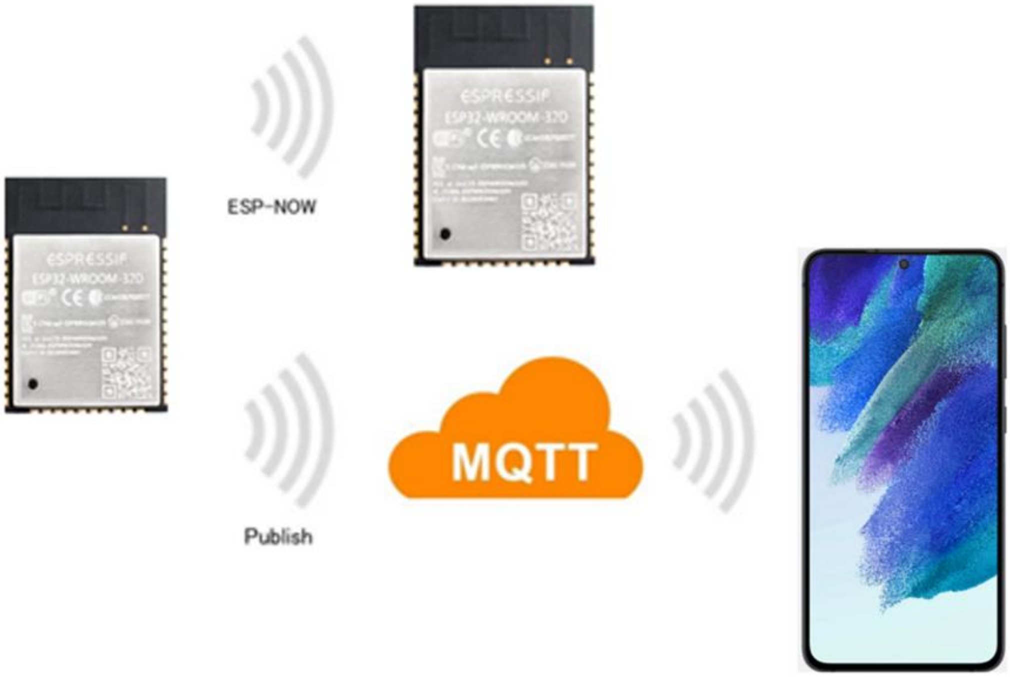


Figure 13 - Protocols working together

3.4 Issues regarding tools and HW/SW aspects

While deciding the different components, there were some issues found. To begin with, it was planned to go with the NCN5150 as it was an alternative to the TSS721A as it was out of stock at the time. After deeper research and testing, it was determined that the NCN5150 delivers less power than the TSS721A. This difference is enough to make the circuit unstable so that the ESP32 does not get enough power delivery to both send data through Wi-Fi, while also processing the data received from the HAN-port. Therefore, the TSS721A gives more stability through more power delivery. The TSS721A was also recommended by our Advisor making it a clearer choice.

It was originally planned to use Multisim as the main circuit design program. However, the lack of supported components quickly led us away from this program. Other applications were also considered, but ultimately led to the same problem. An example here is OrCAD [22] and KiCad [23]. Ultimately EasyEDA was the only software solution that had our components in its database.

3.5 Problem analysis conclusion

To make the chapter more readable and understandable, the subheadings are divided into hardware and software.

3.5.1 Hardware

As a conclusion to the problem analysis, it has been decided that the ESP32 will be the main processing unit. This opens for the use of C/C++ and the Arduino IDE, a programming language and compiler that the group is familiar with and ready to work with. As for the rest of the components, for the conversion from M-BUS to TTL, the group decided to go for the TSS721A transceiver. This component is chosen over the NCN5150 because of its superior power delivery to the ESP32.

For the converters from 36V to 3.3V and 5V to 3.3V we decided to stick with the LT1936 and the R-783.3-1.0 respectively as the components are a perfect fit for this project. Another benefiting factor is that these components are all parts that are supported by EasyEDA. Making it possible to create a circuit design within one singular program. For the SPD, it is planned to include a display to show data. The display chosen is the ILI9341 2.8-inch display.

3.5.2 Software

There are multiple protocols to consider in this project. The protocols that are directly tied to the project specification are the HAN protocol interface and the MQTT protocol. MQTT allows us to publish and subscribe to topics using Wi-Fi, while the HAN protocol allows us to collect power data from AMS readers. A protocol that we decided to add that was not a part of the specification is the ESP-NOW protocol. The reason for the addition of ESP-NOW is for ease of use for the general consumer, as well as resulting in overall lower power draw.

Next to EasyEDA and the Arduino IDE, it was determined to also use Azure Functions through Visual Studio because of the practicality of their cloud-based services where the cost scales with on demand requests. Since Azure Functions supports C# and our group has good knowledge of this object-oriented language, it ended up being a clear choice.

4 Realization of selected solution

The illustration below shows a block diagram of the complete realization of the solution. It combines backend, frontend and the hardware aspects of the solution and shows everything together categorized.

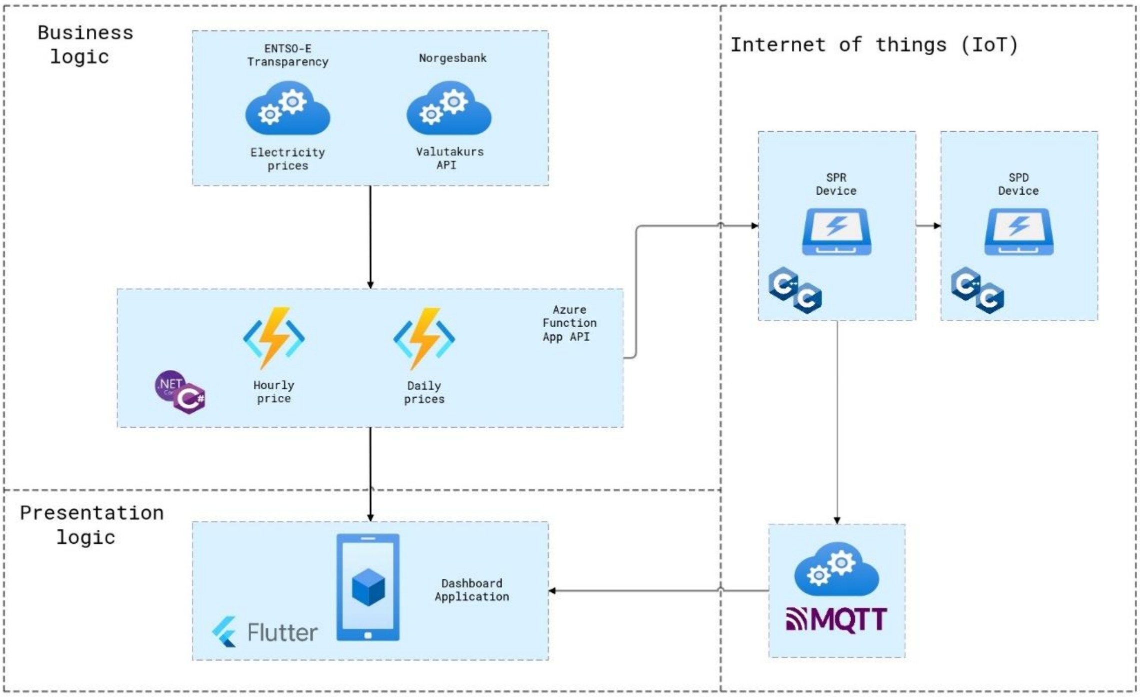


Figure 14 Block diagram of complete solution

4.1 Hardware analysis

Creating a circuit requires lots of preparation. Therefore, creating diagrams and following the datasheets given is critical for a successful project. Thus, digging deeper into each will give more clarity to the project.

4.1.1 Circuit block diagram

An organized and systematic way to visualize a circuit is using block diagrams. It lays out a solid foundation to follow onward as the circuit is being designed. The block diagram in Figure 15 shows the entire circuit. The idea was to have a circuit that is suitable for both the SPR- and the SPD device. Meaning that both devices use the same circuit but have different enclosures that separate them. This opens for faster and better testing experience, while also reducing cost significantly.

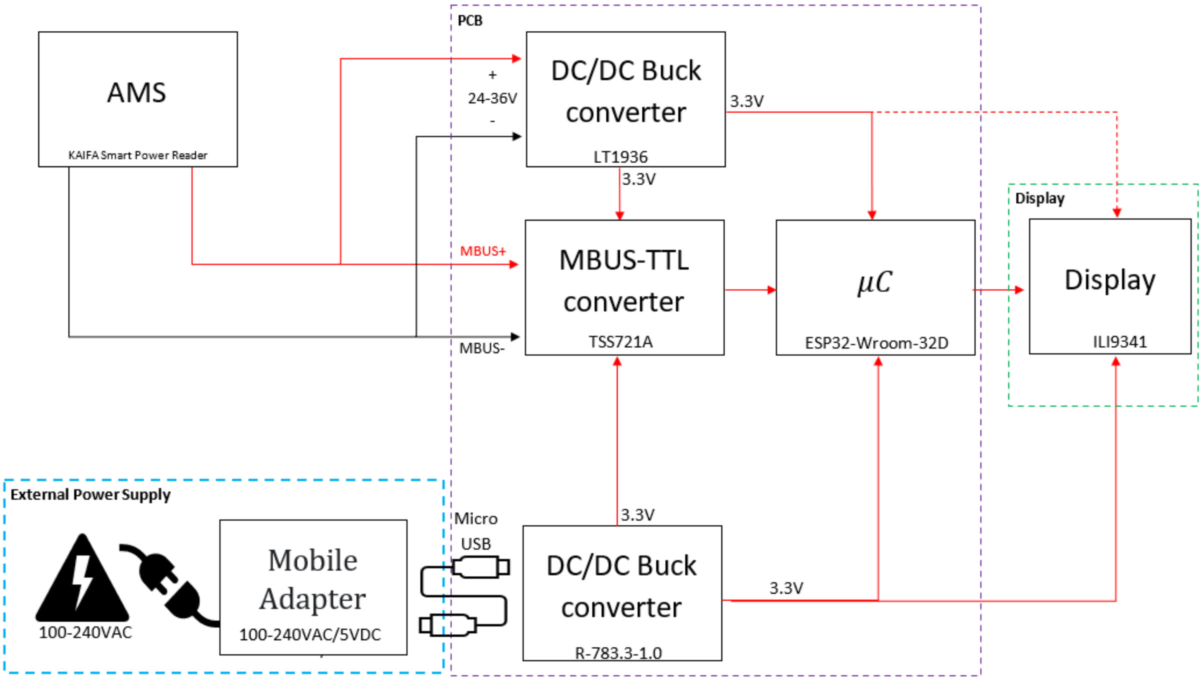


Figure 15 - Complete circuit block diagram

Although the SPR and the SPD use the same circuit board, their functions on the circuit board can be divided into two separate block diagrams as seen in Figure 16 and Figure 17. Both diagrams will each have its own rundown in chapter 4.1.3 and chapter 4.1.4.

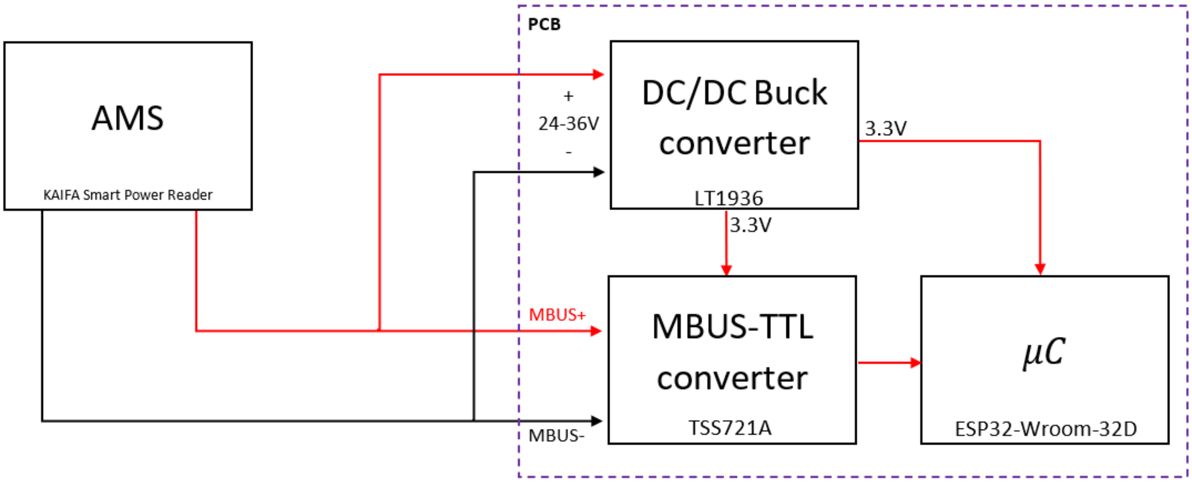


Figure 16 – SPR block diagram

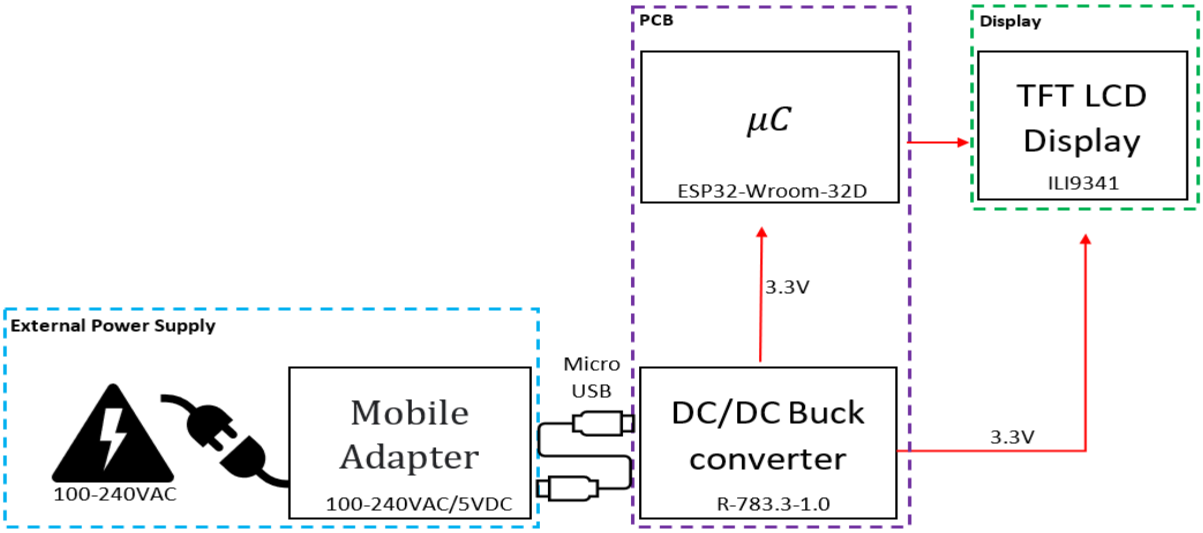


Figure 17 - SPD block diagram

4.1.2 ESP32 in the SPR and the SPD

The ESP32[11] is an integral component to both the SPR and the SPD. It is a feature-rich microcontroller that has low power consumption and supports both Wi-Fi and Bluetooth operations. What is more, the ESP32 has a wide operating temperature from -40 °C to +125 °C with the recommended operating temperature being up to +80 °C, making it robust and highly compatible with many smaller circuits. The ESP32 additionally has plenty of General Purpose Input Output (GPIO) ports which opens for future expandability.

As said, the shared circuit includes ESP32 as the main microcontroller unit. The microcontroller features 4 MB of integrated SPI flash, an onboard antenna, and a small, compact design. With all its features, the ESP32 needs an average of 80mA operating current, and a minimum of 500mA of current delivery by a power supply. This is no issue for the SPD as it has an external 5V power supply, however for the SPR it proved to be more difficult. The SPR only gets its power delivery through the HAN interface, which does not support power delivery over 500mW based on the HAN documentation, however the interface has been tested to be able to deliver over 700mW. Still, it was our mission to have as low power consumption as possible therefore the ESP32 was the perfect microcontroller for the job as it support the low power ESP-NOW protocol as well. However, even with its low power consumption, the ESP32 still needs a buck converter to operate in a stable manner, hence the LT1936.

4.1.3 SPR circuit breakdown

As said, the complete circuit block diagram can then be separated into two independent diagrams. The first diagram (Figure 16) to be explained is the SPR unit. Here we can see that only the LT1936, TSS721A and the ESP32 microcontroller are necessary to keep the SPR functional. The SPRs main function is to read the M-Bus signals sent from the AMS reader, get power delivery from the AMS reader, and finally process the information given and send it to the cloud. With this in mind, an explanation as to why we chose the given components is in order.

In summary, these are the hardware requirements the SPR has:

- Convert 36V-24V down to 3.3V
- Single cable (RJ-45) power delivery from the HAN-port
- Convert M-BUS data from the HAN interface to TTL

- Use the onboard antenna to communicate via Wi-Fi
- Use ESP-NOW to send data with the use of the ESP32 microcontroller

4.1.3.1 LT1936 Step-Down converter

The LT1936 is a current mode PWM step-down DC/DC converter. What is more, the LT1936 also has an internal 1.9A power switch. The converter has nearly 84% efficiency for 3.3V/1.2A output shown at Figure 18. With an input range of 3.6V to 36V the converter is well-suited for our project. The circuit diagram in Figure 19 shows the simple build up for a 3.3V step-down converter found in the LT1936-datasheet [17].

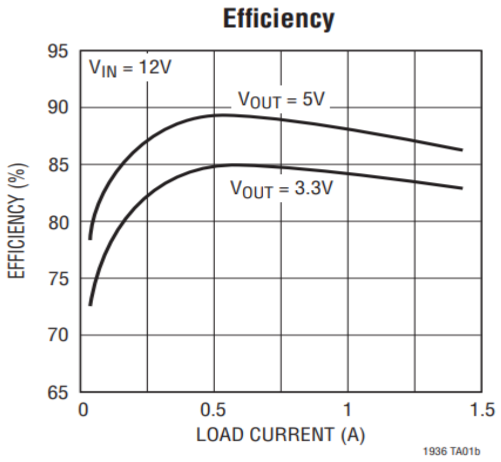


Figure 18 - Efficiency graph for 3.3V/1.2A

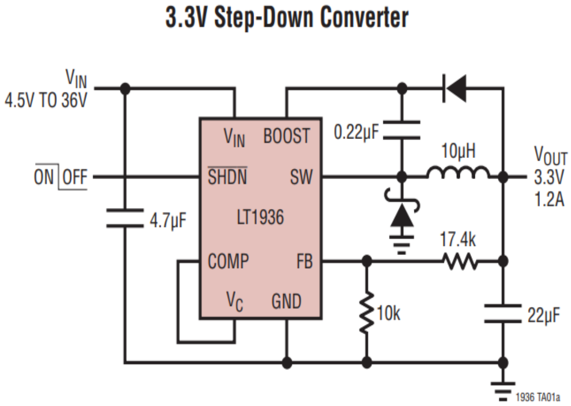


Figure 19 - Step-Down Converter to 3.3V circuit

To understand the step-down converter, here is a breakdown of the most important pins:

The **BOOST** pin is used to deliver a drive voltage, which is higher than the input voltage, to the internal bipolar NPN power switch. The BOOST pin must be at least 2.3V above the SW pin for best efficiency. Hence choosing appropriate capacitor and diode connecting to the BOOST pin is necessary.

The **SW** pin is the output of the internal power switch. This pin connects to the inductor, catch diode and boost capacitor

The **$\overline{\text{SHDN}}$** pin is used to put the LT1936 in shutdown mode. To shut down the LT1936, tie the pin to ground, however if normal operation is wanted, tie the pin to 2.3V up to 5V.

The **Exposed Pad** must be soldered to the PCB and electrically connected to ground. For better thermal performance, a large ground plane and thermal vias was used. More detail is found in the datasheet page 6 [24]

To optimize the performance a step-down converter can provide, we follow as much as we can from the datasheet page 12 [25] and 14 [26], for our circuit and PCB layout.

Inductor

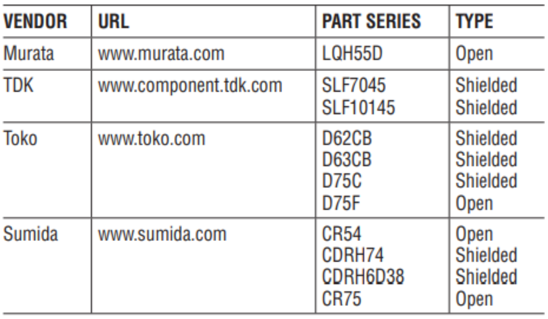
According to the datasheet on page 8 [27], Selection of an inductor value can be simply calculated using the given formula:

$$L = 2.2(V_{OUT} + V_D)$$

From which the output voltage V_{OUT} is to be 3.3V and V_D is the voltage drop of the catch diode (~0.4V) and L is in μH. Which gives us the value of inductor as:

$$L = 2.2 \cdot (3.3 + 0.4) = 8{,}14\ \mu H$$

Based on the value given, we can then look for our inductor by the table of inductor vendors provided on the same page (Figure 18).



VENDOR	URL	PART SERIES	TYPE
Murata	www.murata.com	LQH55D	Open
TDK	www.component.tdk.com	SLF7045 SLF10145	Shielded Shielded
Toko	www.toko.com	D62CB D63CB D75C D75F	Shielded Shielded Shielded Open
Sumida	www.sumida.com	CR54 CDRH74 CDRH6D38 CR75	Open Shielded Shielded Open

Figure 20 - Inductor Vendors

Since we use EasyEDA for our schematic and PCB design, there are only a few components listed on Figure 17 that are currently in stock supplied by JLCPCB [28]. After hours of comparing, we decided to use CDRH6D38NP-100NC [29], manufactured by Sumida Corporation [30]. The CDRH6D38NP-100NC gives a wide range of inductance around $10 \pm 30\%\ \mu H$ and a saturation current & temperature rise comparison graph on Figure 18, which is a perfect match for our design.

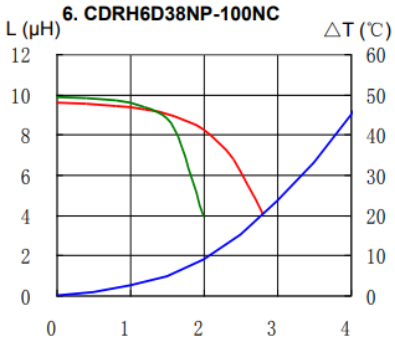


Figure 21 - Saturation Current & Temperature Rise Graph

Catch diode

A recommendation from the datasheet is to have a 1A Schottky diode for the catch diode D1. The diode must have a reverse voltage rating equal to or greater than the maximum input voltage. DFLS140LQ diode [31] is mentioned in the datasheet, it has a reverse voltage rating equal to 40V which is greater than our input voltage and a 1.1A average current.

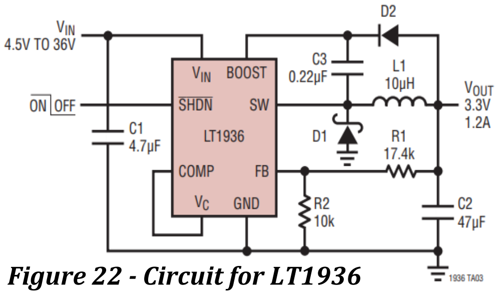


Figure 22 - Circuit for LT1936 buck converter

Boost capacitor and diode

Capacitor C3 and diode D2 on Figure 19 are used to generate a boost voltage that is higher than the input voltage. It is recommended to use $0.22\ \mu F$ capacitor and fast switching diode namely 1N4148W [32].

Input capacitor

Current input supply is drawn in pulses with a fast rise and fall time in Step-Down regulators. The job of an input capacitor is to reduce voltage ripple at the LT1936 and to force this very high frequency switching current into a tight local loop, reducing Electromagnetic interface (EMI). Based on the datasheet, a 4.7μF or higher value ceramic capacitor of X7R or X5R is recommended.

Output capacitor

The function of an output capacitor C3 can be divided into two parts. Firstly, along with the inductor L1, a square wave generated by the LT1936 is filtered to produce the DC output. It determines the output ripple, and for the switching frequency it is important to have a low impedance. Secondly, it stores energy for the purpose of sustaining transient loads and stabilizing the LT1936's control loop.

Protective diode

Diode D4 prevents a shorted input from discharging a backup battery tied to the output. The diode also protects the circuit from a reversed input.

In our case, we use it to prevent a reversed input since we do not have a backup battery.

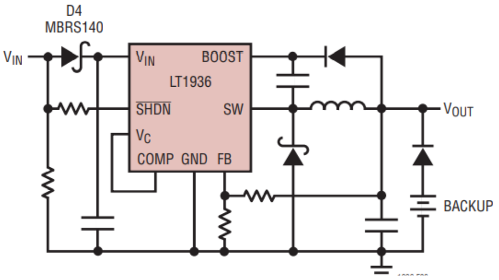
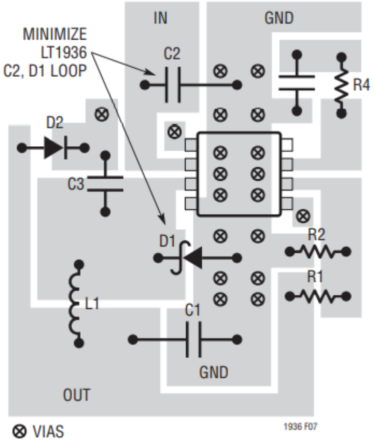


Figure 23 - Protective Diode D4 placement

PCB Layout

Laying out components on a PCB also plays a big part in the project. We place the components as similar as what a recommendation is found in the datasheet. Figure 24 and Figure 25 represent the recommendation and our PCB layout respectively.



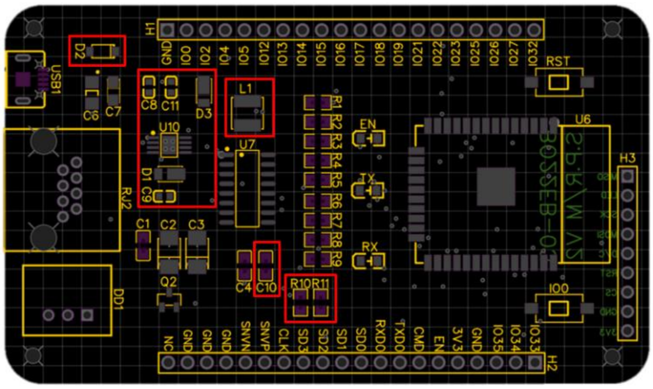


Figure 24 - PCB layout ensures LOW EMI Operation *Figure 25 - PCB layout based on recommendation*

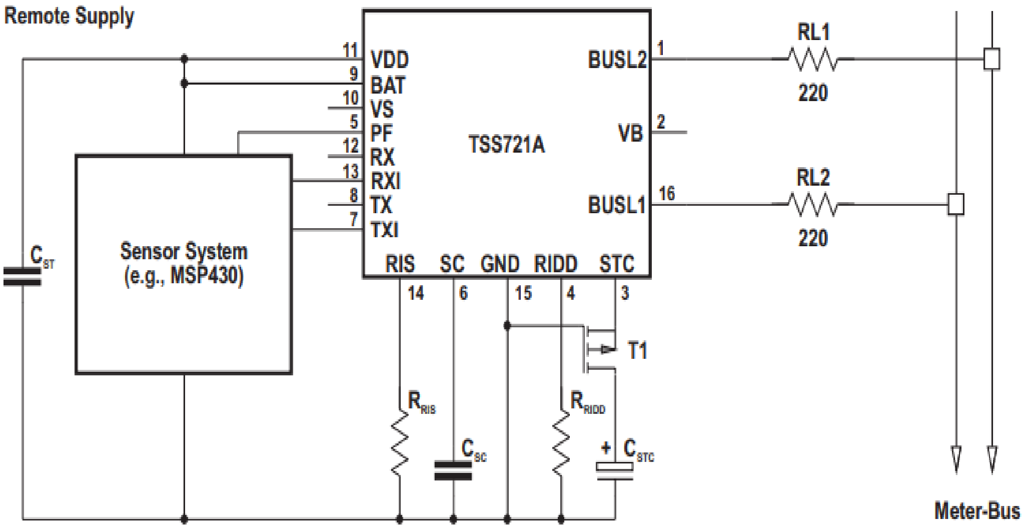
Figure 24 shows the recommended component placement with trace, ground plane and via locations for proper operation and minimize the EMI.

Due to the large, switched currents flow in the LT1936's V_{IN} and SW pins, the catch diode (D1) and the input capacitor (C2). The loop formed by these components should be as small as possible. The Exposed Pad on the bottom of the package must be soldered to a ground as large as it can be so that it acts as a heat sink and to keep thermal resistance as low as possible.

4.1.3.2 The TSS721A M-Bus to TTL transceiver

M-BUS Transceiver is a component which transmits M-BUS data to TTL signal using UART protocol so that both the microcontroller and the computer can understand. The information sent from the HAN interface and out on the "Two-Wire M-Bus" will be marked with an OBIS code, which is an identification of the value that is sent by the smart meter. These OBIS codes will be equal on all AMS meters installed, to ensure that customers can use the same reading device regardless of AMS meters used. To make the data readable for a microcontroller, a suitable transceiver is needed for this job. Texas Instruments has qualified the TSS721A transceiver. This device was redesigned to apply to the EN1434-3 requirements.

TSS721A is a M-BUS to TTL designed by Texas Instrument that only requires few external components like resistors, capacitors, and transistors for operating the component. Figure 23 is found in TSS721A-datasheet under Section Header "Application information" on page 9 [33], we can easily use the block diagram for designing the schematic for us purposes.

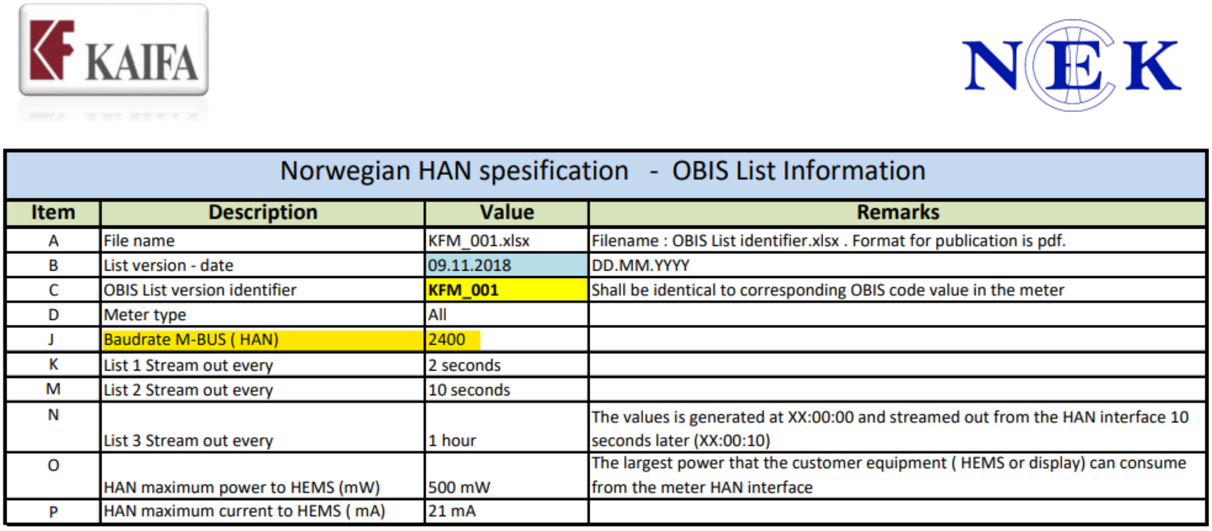


R_{RIDD} = 30 kΩ	C_{STC} =< 220 µF	single load 1UL
R_{RIDD} = 13 kΩ	C_{STC} =< 470 µF	double load 2UL

NOTE: Transistor T1 should be a BSS84.

Figure 26 - TSS721A component block diagram

The transceiver supports up to 9600 Baud in Half Duplex for UART Protocol. According to the KAIFA-KFM_001 specification shown in Figure 24 AMS smart power meter sends a Baud rate of 2400, therefore TSS721A is suitable for the operation.



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

Figure 27 - HAN OBIS list information

Protective Resistors

The bus interface involves two wires, and all slaves connected to the bus can operate only with power provided by the master. Aiming to maintain correct operation of the BUS line, it must have a protective resistor with a value of $(430 \pm 10)\ \Omega$ in the bus line. The bus can reach voltages up to 42 V and the purpose of the protection resistor limits the current in the case of short circuit to a maximum of 100 mA (42 V / 420 Ω). Since the bus are polarity independent and for protecting the slave, the value of the protective resistors (R1 and R2) can split into two which will be 220 Ω max for each line of BUS.

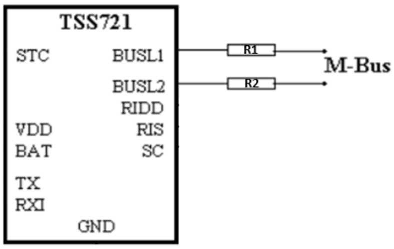


Figure 28 - Protective Resistors R1 & R2

Sampling capacitor

According to the schematic found on Github, designed by roarfred [34] and gskjold [35], a 100nF Sampling capacitor C_{SC} at pin SC is being used. C_{SC} is charged and discharged with a current $I_{SCcharge}$ and $I_{SCdischarge}$ respectively. There must be sufficient time to recharge the capacitor C_{SC} between the bits arriving.

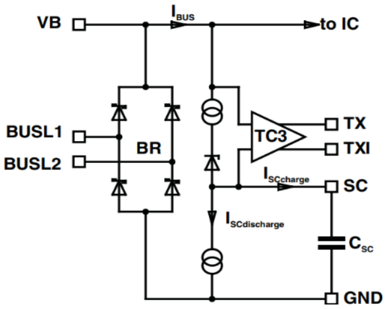


Figure 29 - Sampling capacitor Csc

Storage capacitor

The storage capacitor C_{STC} at pin STC is charged with constant current from the current source CS1. The charge current I_{STC} must be defined by an external resistor R_{RIDD} at pin RIDD. According to the datasheet, the resistor value R_{RIDD} can be chosen from 13k to 80k Ω. The lower the value is chosen, the faster the storage capacitor is charged. We decided to use 22k Ω , the same as the designer Roarfred has.

4.1.4 SPD circuit breakdown

The second figure shows the block diagram for the SPD unit. Here we can see that the ILI9341, R-78 series buck converter and the ESP32 microcontroller are necessary to keep the SPD functional. The SPD in its essence serves as a display that is powered using a 5V Micro-USB cable that recieves data from the SPR and displays said data accordingly.

In summary, these are the hardware requirements the SPD has:

- Convert 5V down to 3.3V
- Receive power delivery using a Micro-USB cable
- Receive data using ESP-NOW with the use of the ESP32 microcontroller
- Use received data and display said data on the screen accordingly

4.1.4.1 TFT LCD Display (ILI9341)

ILI9341 is an LCD display driver that runs on 2.5V to 3.3V. The display has a resolution of 240x320 with a 2.8-inch display.

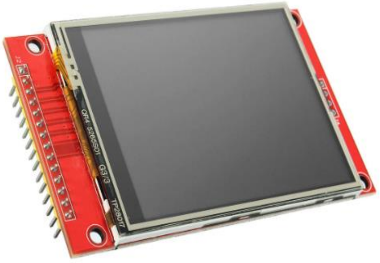


Figure 30 - The ILI9341 2.8-inch display

4.1.4.2 External Power Supply

For the slave device (SPD) to operate, it is required to run with an external power supply. This power supply must give enough power to both the microcontroller and the ILI9341 Display to work. A good example of an external power supply that can be used is a mobile adapter, which already has a buck converter which converts 100-240VAC to 5VDC.

Figure 31 - External power brick

4.1.4.3 5VDC-3.3VDC buck converter

In consideration of providing 3.3VDC to the microcontroller and the display by using external power supply, a second step-down conversion is necessary. Hence, a "R-783.3-1.0" buck converter in Figure 32 is here to take the responsibility.

Figure 32 - 3.3V Buck converter

R-783.3-1.0 has high efficiency of up to 97% shown in Figure 33, while the other 3% get dissipated as heat. It also has significantly low ripple and noise figures shown in Figure 34

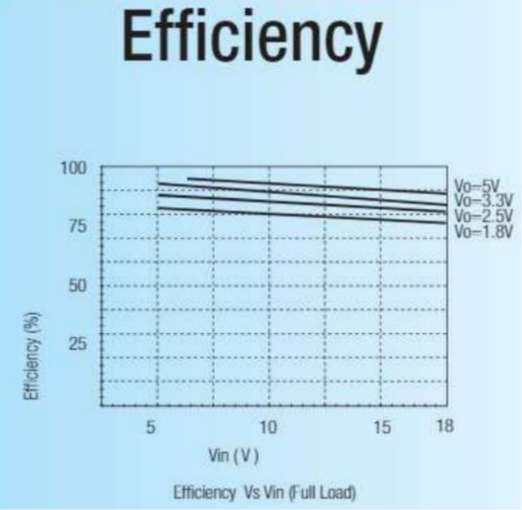


Figure 34 - Efficiency graph

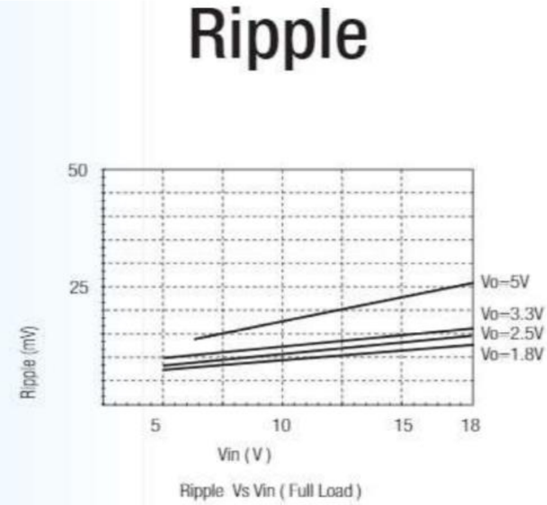


Figure 35 - Ripple graph

The figure on the right is a standard configuration of the 5V to 3.3V buck converter circuit. This is also the circuit that was mainly used on the SPD.

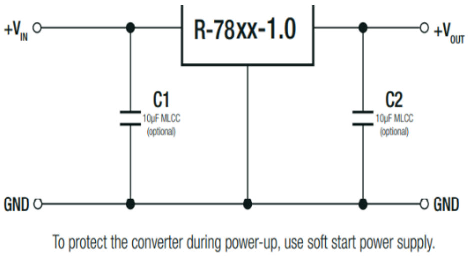

+V_IN
R-78XX-1.0
+V_OUT
C1
10µF MLCC
(optional)
C2
10µF MLCC
(optional)
GND
GND
To protect the converter during power-up, use soft start power supply.

Figure 33 - Buck converter configuration

4.2 Software analysis

To separate the two main software parts, the software subcategories are divided into embedded software overview and software overview. The embedded software portion will explain the software within the SPR and the SPD. The software overview will however include information about Azure functions development, C# development and flutter application development.

4.2.1 Software principles

While creating the code for this project, there were certain programming principles to keep in mind. It is important to keep the code simple and understandable so that other developers can understand and continue developing the code. In addition, comments are spread through the code to explain its purpose and role. This makes it easier to understand and develop the code at a later point.

Additionally, if all the code were to be on one singular file, it would be long and hard to understand. Since among other languages, we are using C++ and C#, two object-oriented languages, using inheritance makes the code readable, trackable, and understandable.

4.2.2 Learn GitHub

While scouring the internet for relevant inspiration, learning how to use GitHub is crucial as a software developer. Learning about branches, forks and other aspects of GitHub is crucial to be able to navigate the internet efficiently.

4.3 Embedded software overview

Embedded software is software written to devices not typically thought of as computers, rather, they are known as embedded systems. Within this chapter, the embedded software development is done upon the ESP32 microcontroller within the SPR and the SPD circuit.

4.3.1 SPR software

The SPR software has a certain sequence it follows after powering up the device. In the block diagram, a detailed software overview can be seen, showing the sequence of events mentioned.

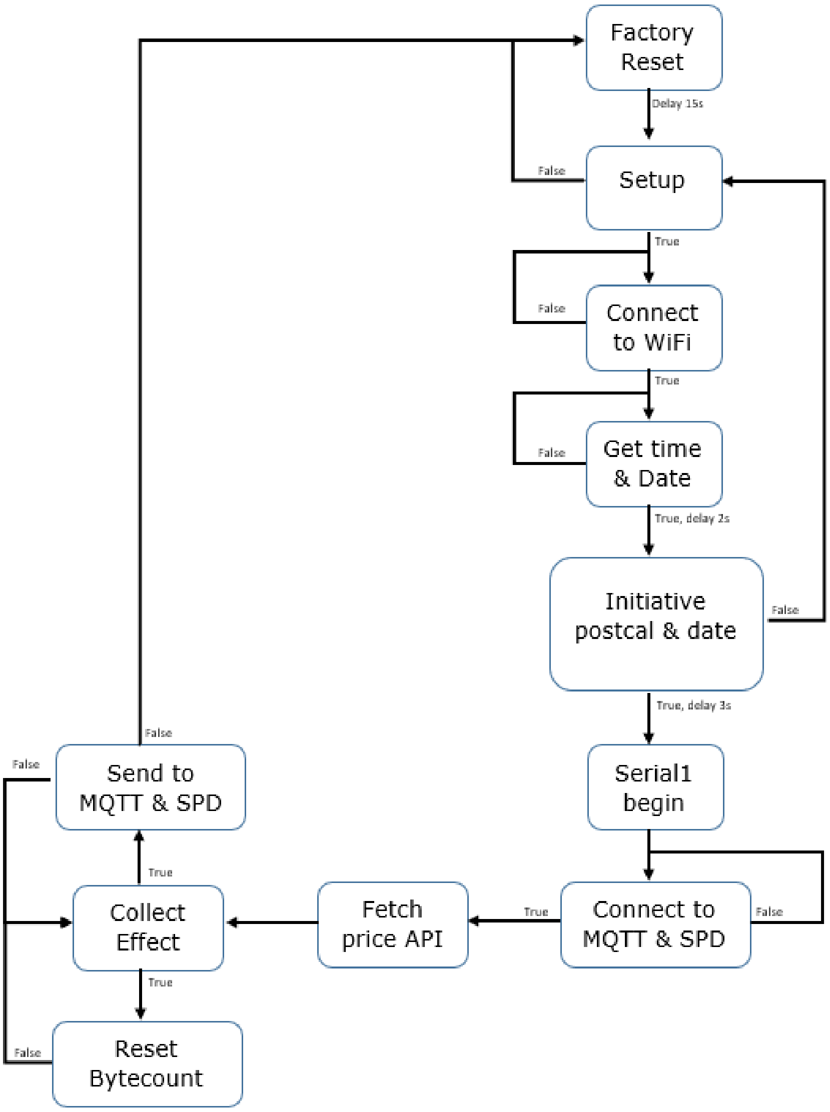


Figure 36 - SPR software block diagram

These are the embedded software requirements the SPR has:

- Create a local access point in which one can connect the SPR to MQTT and local Wi-Fi
- After connecting to Wi-Fi, start sending data to MQTT
- Receive the MAC from access point and connect to SPD using ESP-NOW
- Receive price in NOK using Azure Functions API
- Use ESP-NOW to send price, power usage and power usage per hour to the SPD unit

4.3.1.1 Wi-Fi based operations

The SPR is the main unit, hence explaining what is going on inside the unit brings clarity to its purpose. When the SPR is turned on for the first time, it creates a local access point. This access point then allows the user to connect the SPR to the internet, MQTT, and if necessary to the SPD as well. The option to connect to the SPD using the SPDs MAC Address is optional and can be left blank if it is unwanted. After establishing an internet connection, the SPR then connects to MQTT and starts sending power usage. Additionally, after an internet connection has been established, the spot price is fetched using the Azure Functions rest API once per day. This action is only executed once per day so that as little strain is done to the cloud-based service as possible. Lastly, the SPR fetches the time using an API so that local time is known and used to fetch the spot price in the current date.

4.3.1.2 Data processing

After the SPR has established all its connections, the SPR then starts executing its power reading code. The data is received as TTL and depending on the list length it corresponds to lists one, two or three as seen in Figure 6. After receiving the entire list, the specific bits for power usage then need to be fetched as well. This is done with the use of bit-shifting, taking the first 8-bit value and moving it 8 bits to the left, and adding the second 8-bit to create a 16-bit value that represents power usage.

If the MAC address was received during the access point setup, the SPR will constantly scan for SPD and try to establish a connection with the MAC Address given. If the right MAC Address was given, and a connection was established, the SPR then sends the processed data from the AMS reader to the SPD unit.

4.3.2 SPD software

In the same way as the SPR, the SPD also follows a set sequence of events after the device boots up. In the block diagram, an SPD software block diagram can be seen.

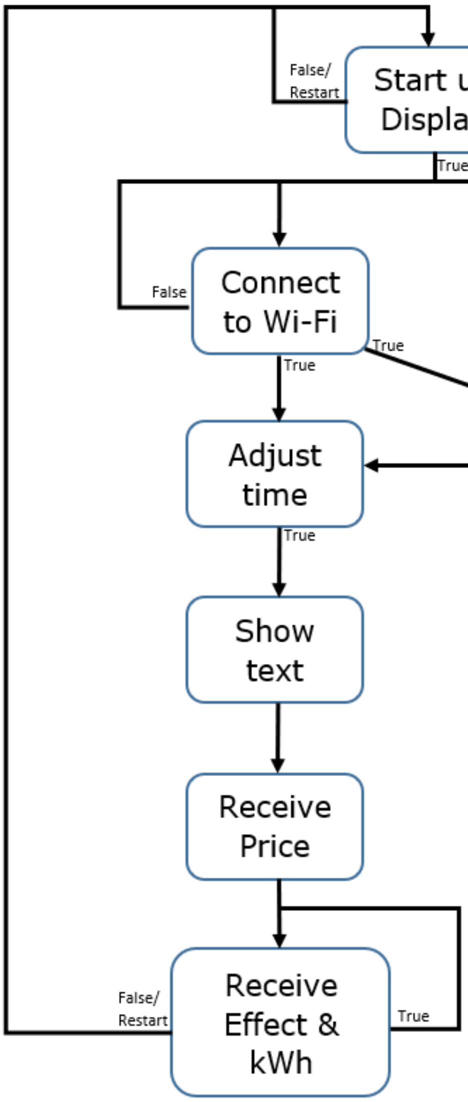


Figure 37 - SPD software block diagram

These are the embedded software requirements the SPD has:

- Create a local access point in which one can connect the SPD to local Wi-Fi
- Receive price, power usage, and power usage per hour using ESP-NOW
- Display power usage, power usage per hour and price on the ILI9341 display

The SPD serves as a smart display, it receives data and displays the given data. However, first a connection needs to be established. The SPD needs to connect to Wi-Fi, the same as the SPR, though the SPD does not need MQTT and MAC Address information for a connection to be established. The SPD in fact has no connection with MQTT to reduce load and power consumption. As for the MAC address, only the sender needs to know this information. Additionally, it fetches current time using a rest API, and displays it on the screen if a Wi-Fi connection has been established.

After the Wi-Fi connection has been established, the SPD then displays its MAC Address on its display and waits for a connection to be established between it and the SPR. After establishing a connection, the SPD shows all relevant information like power usage, power usage per hour and spot price on its display. All data is received as raw data. Power usage and power usage per hour are received as floating points, while the spot price is a float array that consists of twenty-four floating point values that indicate a certain hour's price. All data received is received using ESP-NOW, meaning that no API calls, or MQTT subscriptions are made to receive this data, reducing load and power consumption greatly. As a result, the SPD only needs to receive data, it does not send out any data at all.

4.4 Software overview

The software overview chapter consists of C# software development, and Flutter mobile application development. In the illustration below, we can see business logic, which is logic that happens behind the scenes that the user does not see and presentation logic, which is what a user is presented with. This chapter will mainly feature business logic and presentation logic.

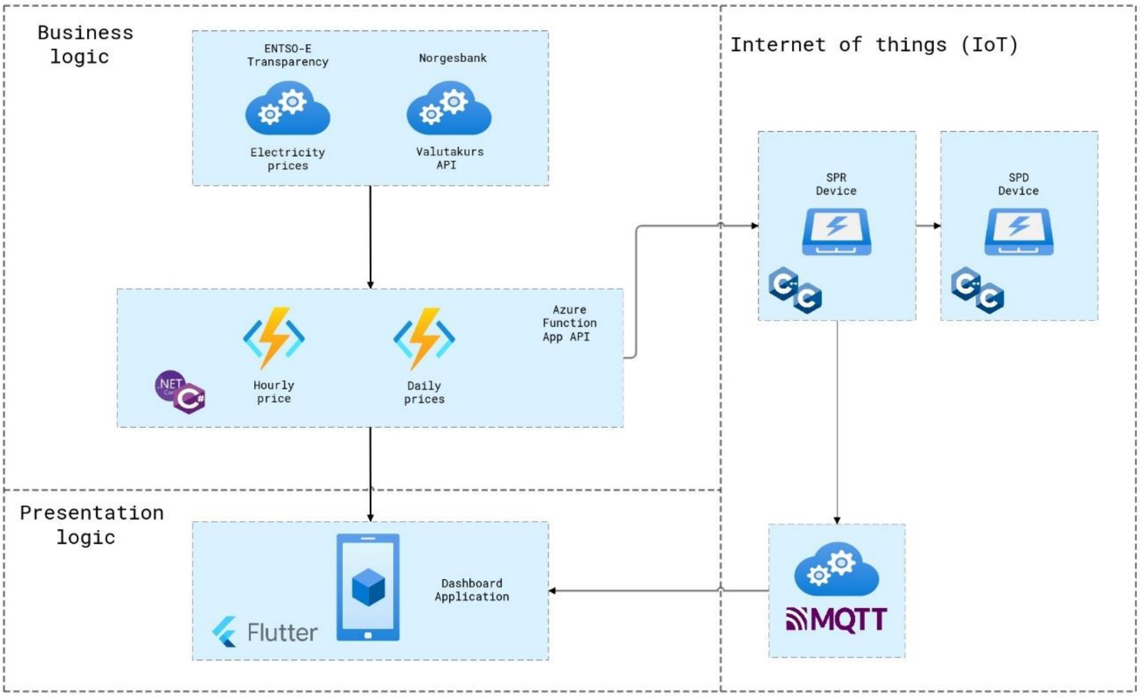


Figure 38 - Software block diagram

4.4.1 C# overview

When working on the software, there were plenty of obstacles to conquer. Firstly, learning Azure trigger-based programming was something none of the team members had done before. After receiving the necessary data as a response message, models then must be made to process the response. This is both for the XML response from Transparency, and the JSON response from Norges Bank. Additionally, converting XML response to JSON in C# had to be solved through extensive research. The solution for receiving these different formats and extracting relevant data is creating models that are specifically made to receive that certain message.

When processing the data and making it ready to send, using a JSON format was chosen when sending the spot price in NOK. The data is sent using dictionaries in C#. This ensures for a key, and a value to be stored. The hours of the day were used as keys, and the spot price were stored as values. This means that the spot price of any hour throughout the day can be fetched by asking for any particular hour, but also the entire dictionary, containing all hours can be sent.

It was also important to “foolproof” the code with use of exceptions, so that if there were to be problems, the exception was to be handled and could be backtraced.

4.4.2 Flutter Mobile application development

Flutter is a cross-platform software development kit designed around the programming language Dart. It is an open-source framework that can develop cross-platform applications using one codebase. Flutter relies on a library of pre-made Material User Interface (MUI) and Cupertino widgets to make a modern and beautiful UI. This makes it simple for people with limited mobile development experience to launch their first minimum viable product.

4.4.2.1 Why chooses Flutter

Flutter is a cross-platform development tool. That means developers can use the same code base for building an app on multiple platforms. From version 2.10 – Flutter supports iOS, Android, Desktop and Web.

What is more, Community support plays a huge role in the development process, because a bigger community means more available resources and third-party libraries. For instance – Syncfusion’s widgets for Cartesian graph and Radial graph are crucial for our application.

4.5 Case design

When it comes to case design there are multiple things to keep in mind. The first thing being the dimensions of the circuit. When designing the circuit, it was important to note that the circuit will later be encapsulated into a case. That is why the circuit was made in a way so that the display can be placed on top of the circuit and connect seamlessly, making it steadfast inside of the case.

Secondly, it was important to design the case in a way that the consumer can hold the device and press reset in a way that feels natural. The thought process here was to be able to hold both the SPR and SPD in a way that is comfortable for the human hand. Even though the product is not meant to be held, having the design in this way opens for future expandability. It is planned to have two holes for interactive buttons on the side of the case; these buttons can in a later stage switch between menus and show the user more metrics. Therefore, we took some inspiration from how controllers and phones are held in a landscape view as seen in the illustration below.

Figure 39-Handheld smartphone in a landscape view

The last criteria concerns the size of the case. The SPD case needs to be wide enough to fit a screen inside it, while keeping as slim of a design as possible, while the SPR needs to be small enough to fit inside of a fuse box. It is also planned to have magnets attached on the back of SPR so that it can be placed to the fuse box door as these are often made of steel.

The program used for this purpose is the AutoDesk Fusion 360 [36]. This program gives free licenses to students and gives the opportunity to design, test and simulate your 3D design. Hence the reason for choosing Fusion 360.

In summary the case design requirements are:

- The case design must fit the circuit and two buttons into it
- The SPR case must fit inside of a fuse box and be as compact as possible
- The SPD case must also fit a display, in addition to a circuit and two buttons
- The fit inside the case must be robust and snug

4.5.1 Design differences between the SPR- and the SPD case

To simplify the design process, a similar design was made for the SPD and the SPR. The main difference is seen on the connectors. The SPD has a connector hole fit for the RJ-45, while the SPR only needs to fit a Micro-USB cable. The other difference is the screen hole and screen mounts necessary for the SPD and its display. Other than that, the case design and dimensions stay the same. This is made possible because of the same circuit board that both devices have.

4.5.2 First case design

Because of the similarity in design, only the SPD case was printed and tested. If the SPD case were to not fit, the same would apply to the SPR, therefore lessening the work necessary for troubleshooting. After the main aspects of the case were rooted, the first case design was ready to be developed. After obtaining access to a 3D printer, we 3D printed the first design. The precise dimensions can be seen in Figure 41.

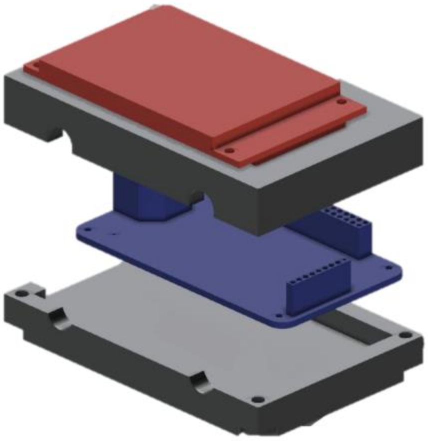


Figure 40 – 3D model of circuit with display

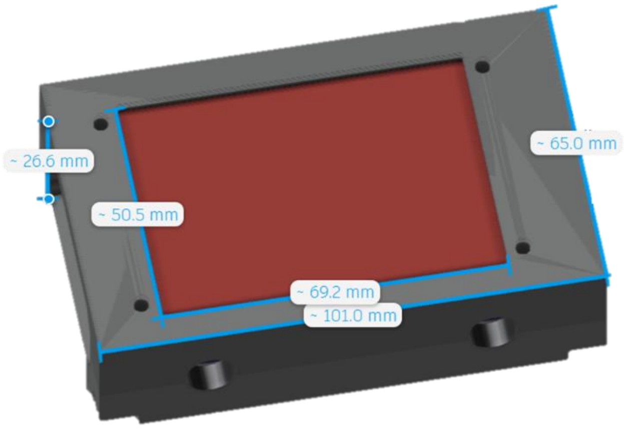


Figure 41 – Case design with dimensions

The first case design ended up being somewhat inaccurate. In Figure 40, the display can be seen as the red part. A slight miscalculation was done here so that there ended up being a slight offset in the case design. The pins at the end of the display were not taken into account. The pins create a raise in the case so that the circuit does not fit into the case. Therefore the design had to be redesigned.

4.5.3 Second case design

4.5.3.1 SPR case

The SPR case features a design that is similar to the first design, however the second design is more symmetrical and only has a port for the RJ-45 cable. Additionally, the circuit is completely enclosed and has two buttonholes on the sides. This design fits finely inside a fuse box, making it meet all the requirements set. The illustrations below show the current SPR case design and its dimensions.

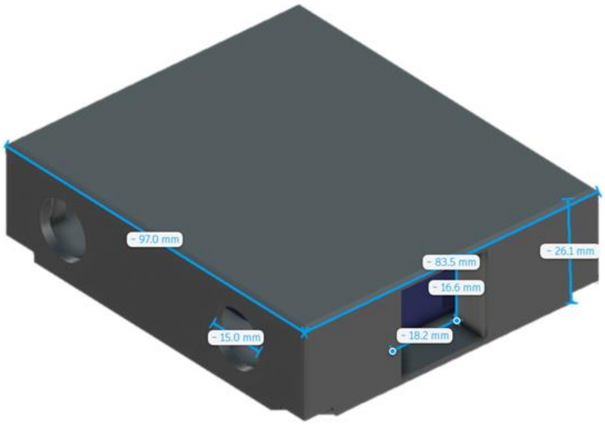


Figure 42 - SPR dimensions

Figure 43 SPR from the side

4.5.3.2 SPD case

The second SPD case compared to the first design, features numerous improvements. The first one being bigger holes that ensure room for the buttons chosen. The next improvement from the first design is that the screen is secured safely from the inside. This means that the mounting holes in the front disappear from the outside. Making the case smoother and better looking. Lastly, the first design had an open side panel where the ports were easily accessible for testing. This left the entire side open, while the latest design only has the relevant ports open. In the SPDs case it is the RJ-45 port that is open from the outside, while the rest is enclosed.

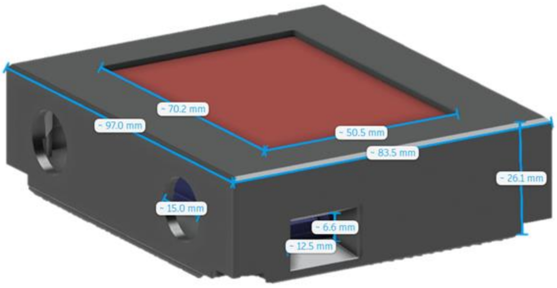


Figure 44 – SPD case design with dimensions

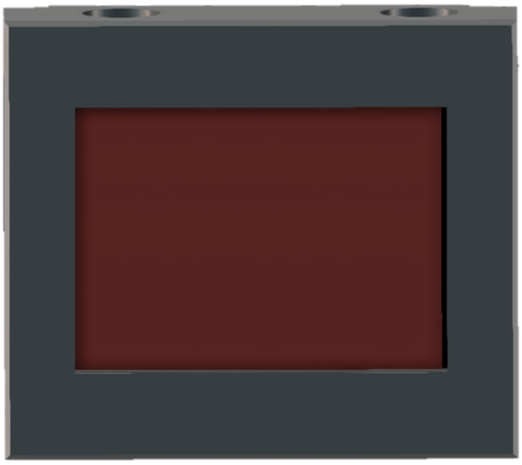


Figure 45 – Case design showing the display

5 Testing

After receiving the components, intensive testing took place to confirm that the various components were functional. When creating a circuit, a lot of things can go wrong. A single mistake can lead to an entire circuit failure. That is why you must be thorough. The tests are divided into three major categories. ESP32 testing, first circuit testing, and second circuit resting.

5.1 ESP32 Thing Plus testing

The ESP32 Thing Plus development board [37] was primarily used as the main board for testing. Before the circuits were designed the code development was still ongoing. This was done on the Thing Plus development card. The upside here is that the software can be developed and tested before the circuit ever was made. Developing in advance ensures for more stability and planning.

5.1.1 Problems encountered

While developing the software for the microcontroller, problems inevitably occurred. The first problem stumbled upon was the problem with integration of ESP8266 code to the ESP32. Even though the two microcontrollers are from the same manufacturer and the same microcontroller-branch, they do not support the same code entirely. With the ESP32 and the ESP8266 being the most popular microcontroller, a lot of their code intertwined on the internet. This made it harder to separate the two as both are written in C/C++. Consequently, some codes that have been taken inspiration from did not work. This is an issue that persisted throughout the project as information on the internet ended up being inconsistent.

Secondly there were multiple crashes happening on the ESP32. To begin with the issue was unknown, as a result, problem solving was needed. To begin with, the ESP32 did not give any reasonable error messages upon failure. This is where printing out status messages throughout the code makes it easier to understand where the program fails, and where the fix needs to be integrated. Through this methodology, we managed to find out what the main issue was. Setting up the Wi-Fi connection saved the information to the flash memory, however if this fails while setting up, the ESP32 might crash as a sub-consequence. This issue was later fixed using a button solution so that a simple press of a button fixes this issue. Though, only through methodical testing was this possible to resolve, otherwise the problem would persist without knowledge of what caused it.

5.2 First circuit

The first circuit ended up as a steppingstone towards a more fulfilled second circuit. Even though the first circuit did not work as intended, a lot was learned through the testing of this circuit as seen below.

5.2.1 Component testing

The first step towards a finished first circuit was ordering components and testing said components. The components that needed testing for the first circuit were the NCN5150, RJ45 Shield and the display. To begin with, the testing of NCN5150 was the most important part. Its job was to convert M-BUS signals to TTL signals while doing so at 3.3V. After testing, it was concluded that the NCN5150 indeed can convert these signals, however, the power supply of this component to the ESP32 was not enough. Therefore, while testing the signals sent to the ESP32, this caused some stability problems causing the ESP32 to shut down while it needed more power.

In addition to testing the NCN5150, the RJ45 shield was tested as a byproduct of the NCN5150s testing as well. This worked as it was supposed to out of the box without fail, hence there being no need to test it further.

Lastly the testing of the display was essential for the SPD unit. The ILI9341 display worked as it should, however its unorthodox library proved difficult in the starting phase. For example, with there not being a *Clear()* function in the library, the screen had to be erased by setting the desired pixels to black, and later overwriting them. This meant that the code for the screen had to be harder than it needs to be, causing repetitive code. However, this is something that was overcome, and the display was later used in the second circuit as well.

5.2.2 Complete circuit testing

After the first circuit arrived, the testing of the circuit could begin. Immediately after the testing had begun, it was noticed that the circuit was not getting enough power. Additionally, through methodical testing of signals to and from the microcontroller, it was noticed that certain connections were not as they should be. Meaning that this was not something that could be changed through modifications to the physical circuit, but rather through a new circuit design.

In other words, after receiving the first circuit from the PCB distributor, the first circuit ended up getting too little power. The NCN5150 did not deliver enough, and the circuit needed a 36V to 3.3V converter to utilize the HAN-ports power delivery. The second circuit also is to include support for a screen, an RJ-45 shield and a Micro-USB shield. Doing this secures the possibility of testing for both the SPR and the SPD at the same time. Making it less expensive budget wise, than ordering two separate circuits to test on, while also making it practical.

5.3 Second circuit

The second circuit led to more success; however, this was only possible through testing. After the extensive testing of the first circuit after it had arrived, the group was ready for the second circuit and had an idea of what was necessary.

5.3.1 Changes from the first circuit

After testing the first circuit, some changes were made to make it a better product. Firstly, the NCN5150 was exchanged with the TSS721A as it now was available for sale again. This also provides more power and stability to the circuit. The second change made was the integration of multiple converters, resulting in more power delivery to the ESP32, and the circuit's components. The converters added are for both 5V to 3.3V and 36V to 3.3V. This means that we can test the power delivery of each and see if it is sufficient to run the circuit independently.

5.3.2 Preliminary component testing

To ensure the stability and safety of the converters, to begin with we started testing on the LT1936 so that we can confirm that it delivers 3.3V/1.2A. The circuit layout suggested in section 4.1.3.1 Figure 18 was then soldered together and tested the claim. After the circuit was put together, using a multimeter the claim was tested to be true and the 3.3V output was achieved.

The next converter tested was the R-783.3-1.0. This converter was tested using an external power supply. The power supply used was a mobile charger that delivers 5V, while the expected output is 3.3V. This was a simpler converter to test, so we quickly noticed that it indeed does give 3.3V as output measured using a multimeter.

5.3.3 Testing results

After the problems with the first circuit, it was necessary to find out if the second circuit would have the same issues. To begin with, the testing of power delivery on the second circuit and it being sufficient was of high priority. The first power delivery source tested was the Micro-USB 5V to 3.3V delivery. This ended up not working to begin with. Additionally, the LT1936 also was not working, so something was wrong. When troubleshooting this issue, we used multimeters and an Agilent Oscilloscope to find out where the problem stems from. After some testing on the circuit, the problem showed itself on the resistors and capacitors on the circuit. With EasyEDA being a newly introduced to software, errors caused by inexperience might occur. What happened is that the resistor values were the same throughout the board, however the names indicating the values were different. Meaning that the value of the resistors and some capacitors was the same, while the name indicated otherwise. This was later fixed by desoldering the previous components and soldering on the new and correct ones.

After this issue was resolved, the circuit worked as intended. The circuit was tested for stability. This proved highly efficient, as it was possible to power the circuit only using the HAN-port power delivery at a constant rate. This was even possible with the display, meaning that future versions of the product can come as an only display option without the need for SPR to send data.

5.3.3.1 HAN interface testing

Receiving data using the HAN interface should now be possible, however there were slight delays. In the beginning we thought that the data was readable straight from the TSS721A without any work needed. The data packets the TSS721A sends are 12-bit packets, however the ESP32 by default does not support this packet size. For the ESP32 to correctly process this information we need to include an important snippet of code, *Serial1.begin(2400, SERIAL_8N1, RX_PIN, TX_PIN)*. This then allows for the ESP32 to process the data and receive said data sent through the HAN interface.

Afterward, we had to find where the relevant data is stored within the packets. With the use of the Agilent oscilloscope, we could visualize and perceive the packages sent. Through the HAN documentation it says that the power usage location in the two second packet is different from the ten second and one-hour packet. After studying data received as well as the documentation, it was managed to obtain the power usage [W] from the two- and ten second packets. While also obtaining the power usage per hour [Wh] from the one-hour packet.

5.3.3.2 ESP-NOW and MQTT testing

After the circuit was made, the already made ESP-NOW and MQTT codes had to be tested on the new circuit. The ESP-NOW protocol connects as it should using the MAC-Address of the receiver as verification. Furthermore, the MQTT connection connects and publishes the data to the cloud without issue, however everything had to be tested to secure the quality of the product.

6 Discussion

In this chapter we will discuss what went well and what went wrong throughout the project.

6.1 What went well

The project itself was a major success. The SPR and the SPD communicate without known issues. All metrics sent from the SPR to the SPD are displayed on the display without issues. Posting data to MQTT and fetching the data to a mobile application also works as intended. Both power usage, and power usage per hour can be shown on the app every two seconds. The second circuit design was a success, and with the right resistors and capacitors, the circuit can be powered with only one cable. The current case design also ensures a snug fit, making the circuit well protected and encapsulated. With all these things going well, the group is happy to call the project a success after accomplishing so much withing such a short time-period.

6.2 What went wrong?

In any product-making project there will always be delays or unanticipated problems. In this case, the project was assigned at a later date than other groups, meaning that there was time pressure for the get-go. As a result, the first circuit was rushed, causing unwanted errors. The circuit designing process took longer than expected. This is both because of stock shortage for the parts we needed, and because of lack of supported circuit design programs at the time. Furthermore, after a lengthy delivery time, the circuit did not work as intended. Thereafter, necessary adjustments had to be made to the circuit to create a second, better version. Since there had to be a second version, the project as a result fell behind schedule. As compensation the bachelor thesis writing was delayed.

The unanticipated delay was software related. When creating the first circuit, there was time pressure to find a fitting circuit design program. It was first planned to use Multisim for this, however, this ended up being incompatible with the components we chose to use. Therefore, searching for an alternative was an unexpected delay. What is more, the circuit also needed the software to be uploaded. This caused more issues than anticipated. The serial data transfer was later solved, however precious time was lost.

Lastly, there were issues regarding the HAN-port and receiving the M-BUS data. This was the most unforeseen problem, as the TSS721A should make the values readable, however, this ceased to be the case. Even though the issue was resolved, the problem cost the project a weeks' time.

7 Conclusion

In conclusion, the project was successful in creating two devices that can communicate with one another using protocols such as MQTT and the ESP-NOW protocol. The two devices, SPR and SPD, can communicate safely with almost no delay. The master device, SPR, can successfully receive and process data delivered from the AMS Reader. This data is later sent to the SPD unit and shown on its display. The display features metrics such as power usage, power usage per hour, spot price and the current time. SPD also features a Micro-USB connection making it possible to have the product anywhere in the household. Moreover, the SPR manages to be running uninterruptedly without the need of an external power supply. The connection is done through a one cable connection that both sends data and delivers power to the components.

The SPR gets the spot price from a rest API through Azure Functions. This data is then processed into an array containing today's price within the entire day. The price is given in NOK as it is converted from its original EUR status using the current exchange rate via Norges Banks API. The current price can also be viewed on a mobile application made using Flutter. The mobile application can through MQTT view the power usage per hour and the power usage right now. Using the same API, the mobile application also shows the current spot price and a graph of the price distribution this day.

The specifications were met and delivered upon even within such a short timeframe. The following are completed tasks.

- Build Complete HW for HAN to Wi-Fi solution
- Make an embedded SW that can receive measurements from the HAN-port and send them via Wi-Fi to a Local server or to a cloud solution
- Fetch spot price call over the internet
- Build a simple display that receive measurement via Wi-Fi and shows it on the display
- Receive measurements and save them on a cloud solution.

The specifications that were not delivered upon are the following:

- Interconnect task 1 & 2 using a cloud solution server

This was a feature that the devices supported to begin with, however, using ESP-NOW proved more power efficient. making the circuit more stable, therefore we progressed away from this specification.

The current circuit works as both the SPR and SPD, however this is not cost-efficient. For future expansion, it is possible to simplify both circuits to fit the needs of the device. Resulting in a cheaper price per unit. Subsequently, making the profit higher as well. For future notice it is possible to simplify the setup of the device even more, making it as user-friendly as possible. In addition, it is possible to improve the application, add more features and make GUI better. These are all options that are available for the future.

References

[1] Western Norway University of Applied Sciences. (n.d.). Høgskulen på Vestlandet - Høgskulen på Vestlandet. Retrieved May 22, 2022, from https://www.hvl.no/

[2] InfoTech AS. (n.d.). Infotech | Komplett leverandør for byggebransjen. Retrieved May 22, 2022, from https://infotech.no/

[3] Fjordkraft. (n.d.). Fjordkraft | hele Norges strømleverandør | Fjordkraft AS. Retrieved May 22, 2022, from https://www.fjordkraft.no/strom/

[4] Tibber. (n.d.). Et uvanlig strømselskap ⚡ Tibber. Retrieved May 22, 2022, from https://tibber.com/no

[5] Homey. (n.d.). *Et bedre smart hjem.* Homey. Retrieved May 22, 2022, from https://homey.app/no-no/

[6] futurehome. (n.d.). futurehome - simpler, safer, smarter home. Retrieved May 22, 2022, from https://www.futurehome.io/

[7] Flutter. (n.d.). Flutter - Build apps for any screen. Retrieved May 22, 2022, from https://flutter.dev/

[8] Arne, V. (2016, June 28). *201603186-1-informasjon-til-kundene-via-han-grensesnittet-i-ams-måleren-obis-koder-1772408_1124902_0.pdf*. Retrieved May 22, 2022, from https://www.nve.no/Media/4307/201603186-1-informasjon-til-kundene-via-han-grensesnittet-i-ams-m%C3%A5leren-obis-koder-1772408_1124902_0.pdf?fbclid=IwAR0_mStq5HAMxgtaLVUX_UEMGNOGqDoahkx5fBQWOj9wJUL4TRTKTqAK4KI#page=1

[9] *BS EN 13757-2:2018 Communication systems for meters Part 2: Wired M-Bus communication*. (2020, February 28). Free Standards Download. Retrieved May 05, 2022, from https://www.freestandardsdownload.com/bs-en-13757-22018-communication-systems-for-meters-part-2-wired-m-bus-communication.html

[10] Kumar. (2022, January 26). UART Communication Protocol - How it works? Codrey Electronics. Retrieved May 01, 2022, from https://www.codrey.com/embedded-systems/uart-serial-communication-rs232/

[11] EasyEDA. (n.d.). EasyEDA - Online PCB design & circuit simulator. Retrieved March 07, 2022, from https://easyeda.com/

[12] Espressif. (2019, May 09). *ESP32-WROOM-32D & ESP32-WROOM-32U*. AIS. Retrieved May 22, 2022, from https://datasheet.lcsc.com/lcsc/1912300803_Espressif-Systems-ESP32-WROOM-32D_C473012.pdf

[13] ON Semiconductor. (2022, January 24). *NCN5150 - Wired M-BUS Slave Transceiver*. NCN5150_D-2316905.pdf. Retrieved May 22, 2022, from https://no.mouser.com/datasheet/2/308/1/NCN5150_D-2316905.pdf

[14] Texas Instruments, Incorporated [SLAS222, B]. (2022, May 11). *TSS721A Meter-Bus Transceiver datasheet (Rev. B)*. tss721a.pdf. Retrieved May 22, 2022, from https://www.ti.com/cn/lit/ds/symlink/tss721a.pdf?ts=1649010999287&ref_url=https%253A%252F%252Flcsc.com%252F

[15] RECOM. (2020, January 15). R-78-1.0-1709942.pdf. Retrieved May 22, 2022, from https://no.mouser.com/datasheet/2/468/R-78-1.0-1709942.pdf

[16] jxchiang. (2014, January 7). ILI9341.pdf. Retrieved May 22, 2022, from https://cdn-shop.adafruit.com/datasheets/ILI9341.pdf

[17] Linear Technology Corporation. (2008, November 18). *LT1936 - 1.4A, 500kHz Step-Down Switching Regulator*. 2006281609_Analog-Devices-LT1936EMS8E-TRPBF_C665573.pdf. Retrieved May 22, 2022, from https://datasheet.lcsc.com/lcsc/2006281609_Analog-Devices-LT1936EMS8E-TRPBF_C665573.pdf

[18] Unicorn Systems a.s. (n.d.). ENTSO-E Transparency Platform. Retrieved May 22, 2022, from https://transparency.entsoe.eu/

[19] Norges Bank. (n.d.). Norges Bank. Retrieved May 22, 2022, from https://app.norges-

bank.no/query/#/en/currency?currency=EUR&frequency=B&startdate=2021-05-22&stopdate=2022-05-22

[20] Ubidots. (n.d.). Ubidots: IoT platform | Internet of Things. Retrieved May 22, 2022, from https://www.ubidots.com/

[21] ESPRESSIF SYSTEMS (SHANGHAI) CO., LTD. (n.d.). Espressif Systems: Overview. Retrieved May 22, 2022, from https://www.espressif.com/

[22] Cadence Design Systems, Inc. (n.d.). PCB Design Software | OrCAD | Cadence. Retrieved May 22, 2022, from https://www.orcad.com/

[23] KiCad. (1992). KiCad EDA - Schematic Capture & PCB Design Software. Retrieved May 22, 2022, from https://www.kicad.org/

[24] Linear Technology Corporation. (2008, November 18). *LT1936 - Pin Function*. 2006281609_Analog-Devices-LT1936EMS8E-TRPBF_C665573.pdf. Retrieved May 22, 2022, from https://datasheet.lcsc.com/lcsc/2006281609_Analog-Devices-LT1936EMS8E-TRPBF_C665573.pdf#page=6

[25] Linear Technology Corporation. (2008, November 18). *LT1936 - Applications Information*. 2006281609_Analog-Devices-LT1936EMS8E-TRPBF_C665573.pdf. Retrieved May 22, 2022, from https://datasheet.lcsc.com/lcsc/2006281609_Analog-Devices-LT1936EMS8E-TRPBF_C665573.pdf#page=12

[26] Linear Technology Corporation. (2008, November 18). *LT1936 - Applications Information - PCB Layout*. 2006281609_Analog-Devices-LT1936EMS8E-TRPBF_C665573.pdf. Retrieved May 22, 2022, from https://datasheet.lcsc.com/lcsc/2006281609_Analog-Devices-LT1936EMS8E-TRPBF_C665573.pdf#page=14

[27] Linear Technology Corporation. (2008, November 18). *LT1936 - Inductor Selection and Maximum Output Current*. 2006281609_Analog-Devices-LT1936EMS8E-TRPBF_C665573.pdf. Retrieved May 22, 2022, from https://datasheet.lcsc.com/lcsc/2006281609_Analog-Devices-LT1936EMS8E-TRPBF_C665573.pdf#page=8

[28] JCLPCB. (n.d.). JLCPCB: PCB Prototype & PCB Fabrication Manufacturer. Retrieved March 19, 2022, from https://jlcpcb.com/

[29] Sumida Corporation. (2017, January 09). *SMD Power Inductor CDRH6D38*. 1809140821_Sumida-CDRH6D38NP-100NC_C167280.pdf. Retrieved May 22, 2022, from https://datasheet.lcsc.com/lcsc/1809140821_Sumida-CDRH6D38NP-100NC_C167280.pdf

[30] Sumida Corporation. (n.d.). SUMIDA CORPORATION. Retrieved April 25, 2022, from https://www.sumida.com/

[31] Diodes Incorporated. (2015, June 17). *DFLS140LQ*. 2004301407_Diodes-Incorporated-DFLS140LQ-7_C526561.pdf. Retrieved May 22, 2022, from https://datasheet.lcsc.com/lcsc/2004301407_Diodes-Incorporated-DFLS140LQ-7_C526561.pdf

[32] JiangSu ChangJiang Electronics Technology CO., LTD. (2015, March 10). *SOD-123 Plastic-Encapsulate Diodes - BAV16W/1N4148W.pdf*. 1809211019_Changjiang-Electronics-Tech--CJ-1N4148W-G_C164915.pdf. Retrieved May 22, 2022, from https://datasheet.lcsc.com/lcsc/1809211019_Changjiang-Electronics-Tech--CJ-1N4148W-G_C164915.pdf

[33] Texas Instruments, Incorporated [SLAS222, B]. (2022, May 11). *TSS721A Meter-Bus Transceiver datasheet (Rev. B) - Application Information*. tss721a.pdf. Retrieved May 22, 2022, from https://www.ti.com/cn/lit/ds/symlink/tss721a.pdf?ts=1649010999287&ref_url=https%253A%252F%252Flcsc.com%252F#page=9

[34] Roarfred. (n.d.). *Electrical Design Schematics*. AmsToMqttBridge. Retrieved March 04, 2022, from https://github.com/roarfred/AmsToMqttBridge/tree/master/Electrical/HAN_ESP_TSS721#pcb

[35] gskjold. (n.d.). *gskjold/AmsToMqttBridge: ESP8266 and ESP32 compatible firmware to read, interpret and publish data to MQTT from AMS electrical meters*

using IEC-62056-7-5 or IEC-62056-21 standard. GitHub. Retrieved March 04, 2022, from https://github.com/gskjold/AmsToMqttBridge

[36] Autodesk Inc. (n.d.). *Fusion 360 | 3D CAD, CAM, CAE & PCB Cloud-Based Software*. Autodesk. Retrieved May 22, 2022, from https://www.autodesk.com/products/fusion-360/overview?term=1-YEAR&tab=subscription

[37] SparkFun Electronics. (n.d.). *SparkFun Thing Plus - ESP32 WROOM (Micro-B) - WRL-15663*. SparkFun Electronics. Retrieved May 22, 2022, from https://www.sparkfun.com/products/15663

[38] Solberg, R. (2019, July 24). *Smart meter part 1: Getting the meter data*. Kode24. Retrieved May 22, 2022, from https://www.kode24.no/guider/smart-meter-part-1-getting-the-meter-data/71287300

Appendix A. Abbreviation

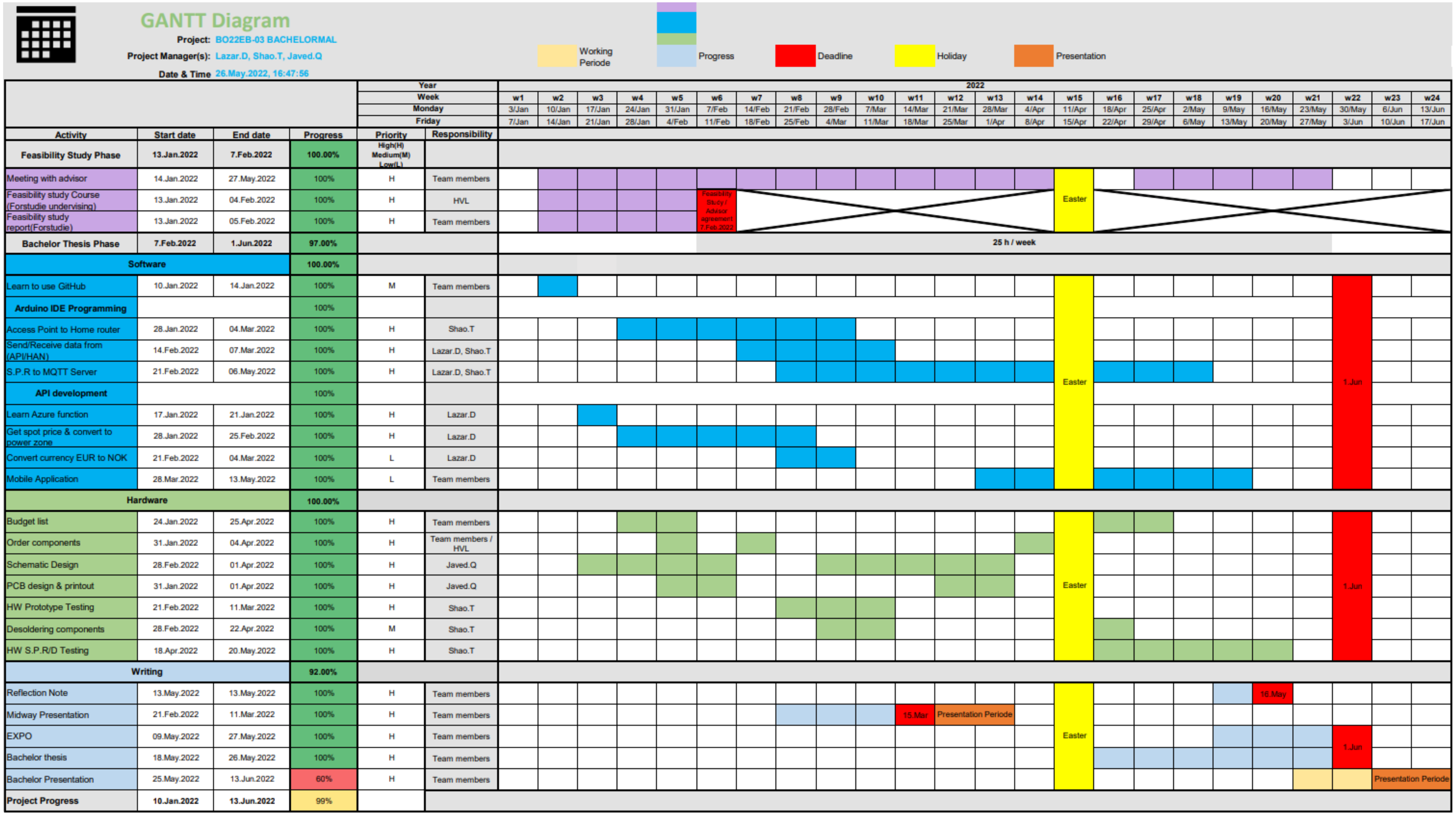
AMS	Advanced Measurement and control Systems
API	Application Programming Interface
CPU	Central Processing Unit
DIY	Do it yourself
EMI	Electromagnetic Interference
ENTSO-E	European Network of Transmission System Operators for Electricity
HAN	Home Area Network
JSON	JavaScript Object Notation
LSB	Least Significant Bit
MAC	Media Access Control
M-BUS	M-Bus or Meter-Bus
MQTT	MQ Telemetry Transport
MUI	Material User Interface
NVE	Norwegian Water Resources and Energy Directorate
PBS	Product Breakdown Structure
PC	Personal Computer
PCB	Printed Circuit Board
SPD	Smart Power Display
SPI	Serial Peripheral Interface
SPR	Smart Power Reader
Transceiver	Transmitter and Receiver
TTL	Transistor-Transistor Logic
UART	Universal Asynchronous Receiver Transmitter
UI	User Interface
USB	Universal Serial Bus
WLAN	Wireless Local Area network
XML	Extensible Markup Language

Appendix B. Datasheet

Component	Designator	Footprint	JLCPCB Part #	Mouser Part #
X5511FV-20-C70D30-1000	H1,H2	HDR-TH_20P-P2.54-V-F-3	C2684741	_
920-C62A2021S10100	USB1	MICRO-USB-SMD_MICRO-5P-5.9	C40939	_
C136740	IO0,RST	SW-SMD_L6.0-W3.5-LS8.0-1	C136740	_
DFLS140LQ-7	D3,D1	POWERDI-123_L2.8-W1.8-LS3.7-RD	C526561	_
RC0805FR-0710KL	R6	R0805	C84376	_
CR0805F82202G	R3	R0805	C101566	_
ESP32-WROOM-32D	U6	WIFIM-SMD_39P-L25.5-W18.0-P1.27-BL	C473012	_
TZ-P2-0805YGTCS2-0.8T	U5,U8,U9	LED0805-RD_YELLOW	C688885	_
1N4148W-G	D2	SOD-123_L2.8-W1.8-LS3.7-RD	C164915	_
CR-05FL7--470R	R4,R5	R0805	C280005	_
CDRH6D38NP-100NC	L1	IND-SMD_L4.0-W4.0	C167280	_
R-RJ45R08P-A004	RJ2	RJ45-TH_R-RJ45R08P-A004	C385834	_
0805B104K500NT	C1,C4,C5,C10	C0805	C38141	_
BSS84LT1G	Q2	SOT-23-3_L2.9-W1.3-P1.90-LS2.4-BR	C82079	_
TC212B227K006Y	C3,C2	CAP-SMD_L3.5-W2.8-R-RD	C110311	_
TAJA106K016RNJ	C7	CASE-A_3216	C7171	_
CR0805J220RP05Z	R11,R10,R1,R2,R7,R8,R9	R0805	C881343	_
T491A106M020AT	C6	3216[1206] - TANTCAP	C117023	_
CL10B224KB8NNNC	C9	C0603	C64705	_
CL10A226MQ8NRNC	C8	C0603	C59461	_
C1608X5R1E475MT000E	C11	C0603	C342979	_
C39576	H3	HDR-TH_9P-P2.54-V-F-1	C39576	_
TSS721AD	U7	SOIC-16_L9.9-W3.9-P1.27-LS6.0-BL	C2871278	_
LT1936EMS8E#TRPBF	U10	MSOP-8_L3.0-W3.0-P0.65-LS5.0-BL-EP	C665573	_
R-783.3-1.0	DD1	R-78		919-R-783.3-1.0

Appendix C. Project management

A.1 GANTT Diagram



GANTT Diagram

Project: BO22EB-03 BACHELORMAL

Project Manager(s): Lazar.D, Shao.T, Javed.Q

Date & Time 26.May.2022, 16:47:56

▨ Working Periode — 🟪🟦🟩🟦 Progress — 🟥 Deadline — 🟨 Holiday — 🟧 Presentation

				Year		2022																							
				Week		w1	w2	w3	w4	w5	w6	w7	w8	w9	w10	w11	w12	w13	w14	w15	w16	w17	w18	w19	w20	w21	w22	w23	w24
				Monday		3/Jan	10/Jan	17/Jan	24/Jan	31/Jan	7/Feb	14/Feb	21/Feb	28/Feb	7/Mar	14/Mar	21/Mar	28/Mar	4/Apr	11/Apr	18/Apr	25/Apr	2/May	9/May	16/May	23/May	30/May	6/Jun	13/Jun
				Friday		7/Jan	14/Jan	21/Jan	28/Jan	4/Feb	11/Feb	18/Feb	25/Feb	4/Mar	11/Mar	18/Mar	25/Mar	1/Apr	8/Apr	15/Apr	22/Apr	29/Apr	6/May	13/May	20/May	27/May	3/Jun	10/Jun	17/Jun
Activity	Start date	End date	Progress	Priority	Responsibility																								
Feasibility Study Phase	13.Jan.2022	7.Feb.2022	100.00%	High(H) Medium(M) Low(L)																									
Meeting with advisor	14.Jan.2022	27.May.2022	100%	H	Team members		🟪	🟪	🟪	🟪	🟪	🟪	🟪	🟪	🟪	🟪	🟪	🟪	🟪	Easter		🟪	🟪	🟪	🟪	🟪			
Feasibility study Course (Forstudie undervising)	13.Jan.2022	04.Feb.2022	100%	H	HVL		🟪	🟪	🟪	🟪	Feasibility Study / Advisor agreement 7.Feb.2022																		
Feasibility study report(Forstudie)	13.Jan.2022	05.Feb.2022	100%	H	Team members		🟪	🟪	🟪	🟪																			
Bachelor Thesis Phase	7.Feb.2022	1.Jun.2022	97.00%								25 h / week																		
Software			100.00%																										
Learn to use GitHub	10.Jan.2022	14.Jan.2022	100%	M	Team members		🟦													Easter							1.Jun		
Arduino IDE Programming			100%																										
Access Point to Home router	28.Jan.2022	04.Mar.2022	100%	H	Shao.T				🟦	🟦	🟦	🟦	🟦	🟦															
Send/Receive data from (API/HAN)	14.Feb.2022	07.Mar.2022	100%	H	Lazar.D, Shao.T							🟦	🟦	🟦	🟦														
S.P.R to MQTT Server	21.Feb.2022	06.May.2022	100%	H	Lazar.D, Shao.T								🟦	🟦	🟦	🟦	🟦	🟦	🟦		🟦	🟦	🟦						
API development			100%																										
Learn Azure function	17.Jan.2022	21.Jan.2022	100%	H	Lazar.D			🟦																					
Get spot price & convert to power zone	28.Jan.2022	25.Feb.2022	100%	H	Lazar.D				🟦	🟦	🟦	🟦	🟦																
Convert currency EUR to NOK	21.Feb.2022	04.Mar.2022	100%	L	Lazar.D								🟦	🟦															
Mobile Application	28.Mar.2022	13.May.2022	100%	L	Team members													🟦	🟦		🟦	🟦	🟦	🟦					
Hardware			100.00%																										
Budget list	24.Jan.2022	25.Apr.2022	100%	H	Team members				🟩	🟩										Easter	🟩	🟩					1.Jun		
Order components	31.Jan.2022	04.Apr.2022	100%	H	Team members / HVL					🟩		🟩							🟩										
Schematic Design	28.Feb.2022	01.Apr.2022	100%	H	Javed.Q			🟩	🟩	🟩	🟩			🟩	🟩	🟩	🟩	🟩											
PCB design & printout	31.Jan.2022	01.Apr.2022	100%	H	Javed.Q					🟩	🟩						🟩	🟩											
HW Prototype Testing	21.Feb.2022	11.Mar.2022	100%	H	Shao.T								🟩	🟩	🟩														
Desoldering components	28.Feb.2022	22.Apr.2022	100%	M	Shao.T									🟩	🟩						🟩								
HW S.P.R/D Testing	18.Apr.2022	20.May.2022	100%	H	Shao.T																🟩	🟩	🟩	🟩	🟩				
Writing			92.00%																										
Reflection Note	13.May.2022	13.May.2022	100%	H	Team members															Easter				🟦	16.May				
Midway Presentation	21.Feb.2022	11.Mar.2022	100%	H	Team members								🟦	🟦	🟦	15.Mar	Presentation Periode												
EXPO	09.May.2022	27.May.2022	100%	H	Team members																			🟦	🟦	🟦	1.Jun		
Bachelor thesis	18.May.2022	26.May.2022	100%	H	Team members																🟦	🟦	🟦	🟦	🟦	🟦			
Bachelor Presentation	25.May.2022	13.Jun.2022	60%	H	Team members																					▨	▨	Presentation Periode	
Project Progress	10.Jan.2022	13.Jun.2022	99%																										

Figure 46 - GANTT Diagram

A.2 Work Breakdown Structure

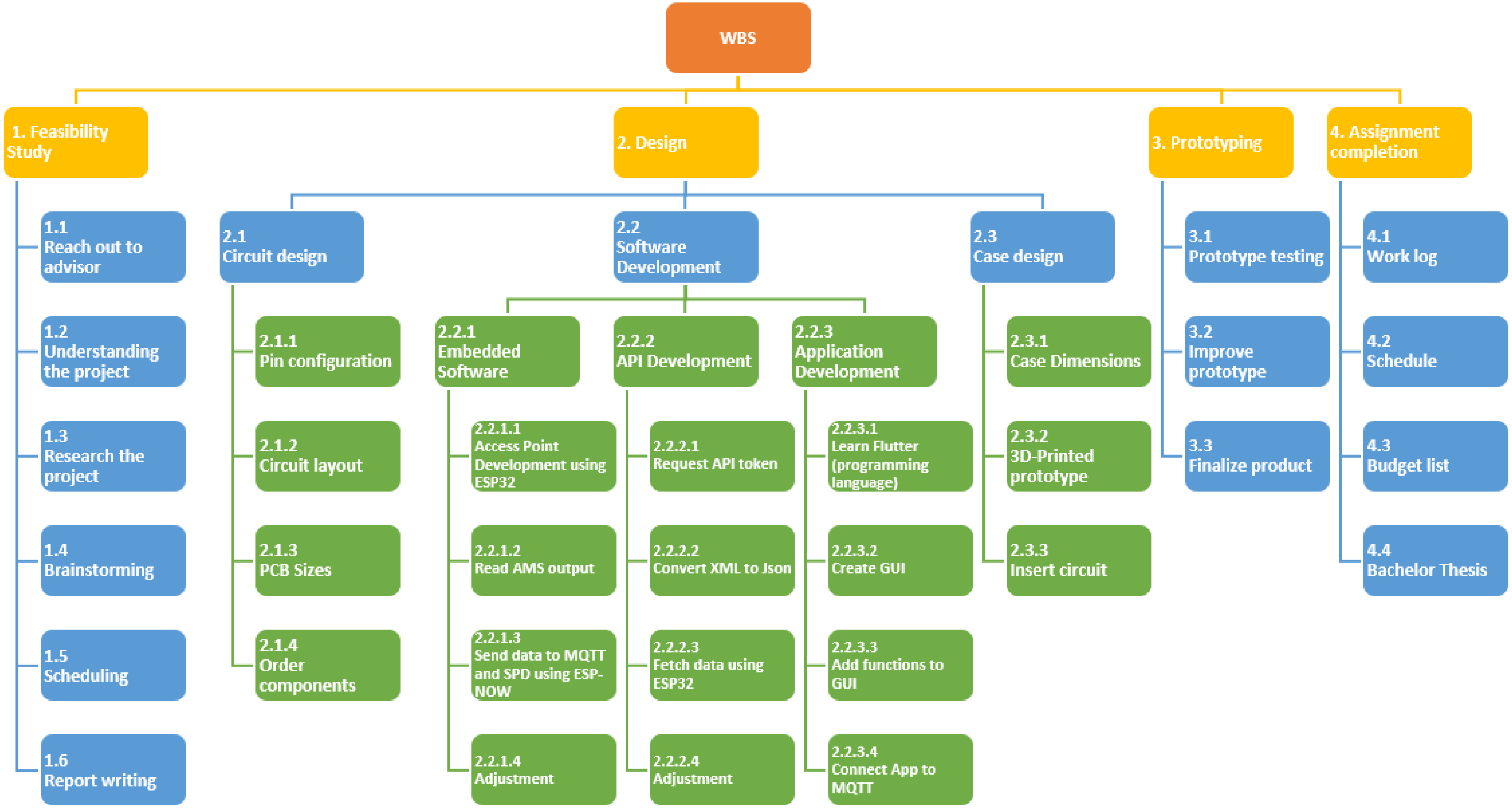


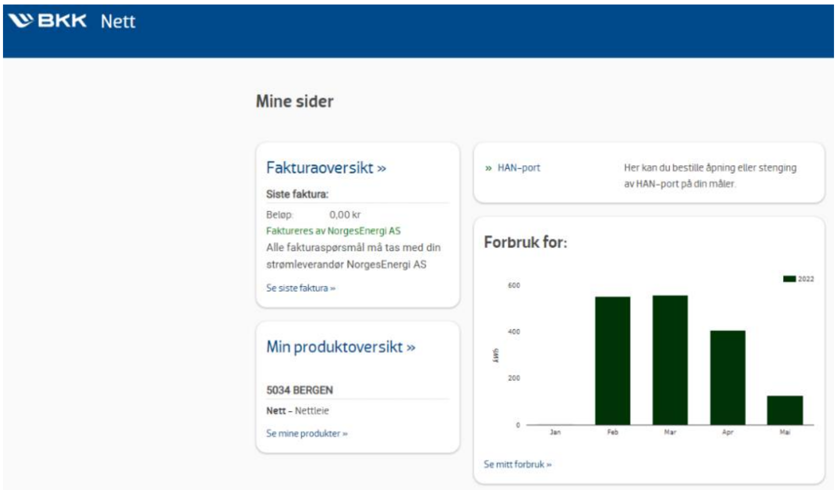
Figure 47 - Work Breakdown Structrue

Appendix D. User manual

A.1 Open HAN port

Below uses **Norway grid company**'s website (BKK AS) as demonstration.

Step 1: Log in to its website (as you are already their customer)



BKK Nett

Mine sider

Fakturaoversikt »

Siste faktura:

Beløp: 0,00 kr

Faktureres av NorgesEnergi AS

Alle fakturaspørsmål må tas med din strømleverandør NorgesEnergi AS

Se siste faktura »

Min produktoversikt »

5034 BERGEN

Nett - Nettleie

Se mine produkter »

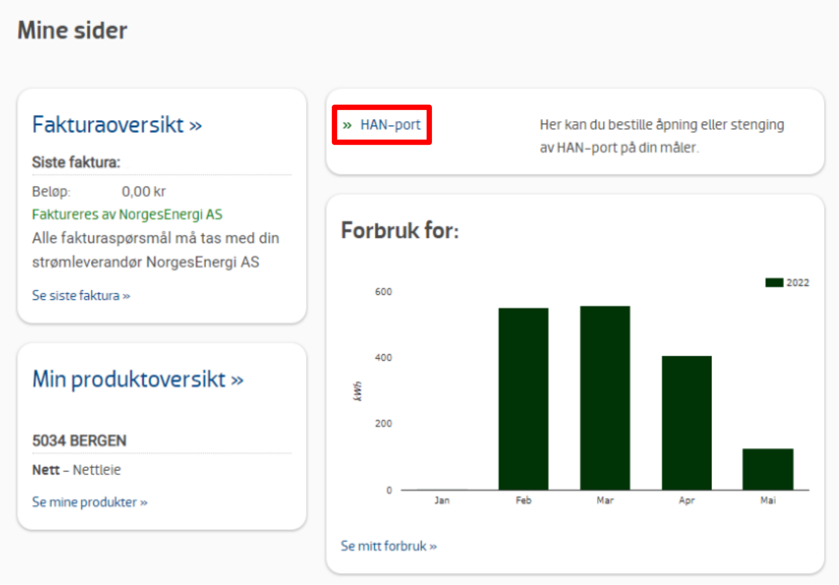
» HAN-port

Her kan du bestille åpning eller stenging av HAN-port på din måler.

Forbruk for:

Se mitt forbruk »

Step 2: Click the link "HAN-port" places on the middle



Mine sider

Fakturaoversikt »

Siste faktura:

Beløp: 0,00 kr

Faktureres av NorgesEnergi AS

Alle fakturaspørsmål må tas med din strømleverandør NorgesEnergi AS

Se siste faktura »

Min produktoversikt »

5034 BERGEN

Nett - Nettleie

Se mine produkter »

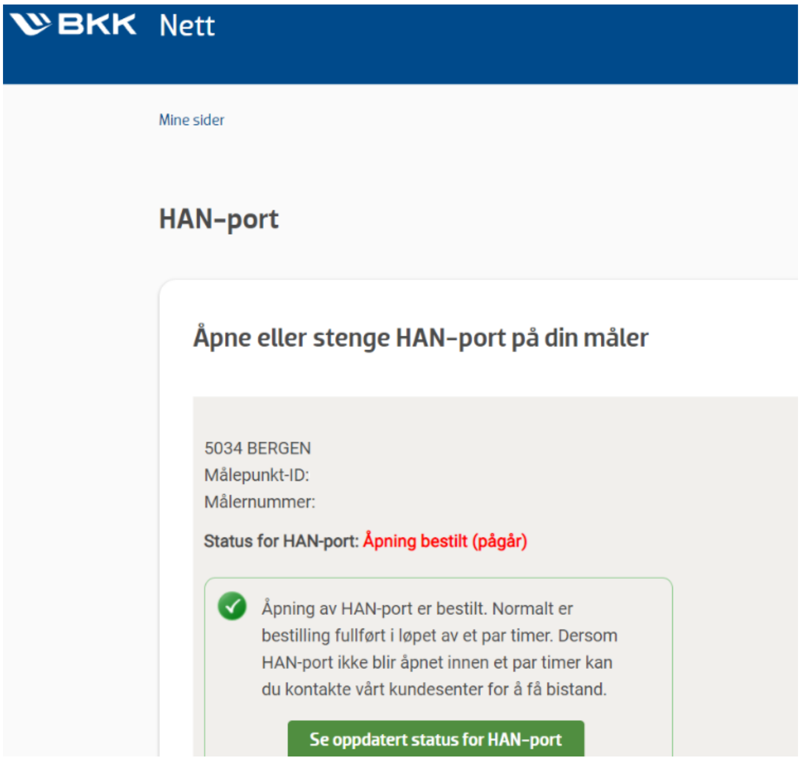
» HAN-port

Her kan du bestille åpning eller stenging av HAN-port på din måler.

Forbruk for:

Se mitt forbruk »

Step 3: Request for opening the HAN-port (figure below shows after requesting)



HAN-port

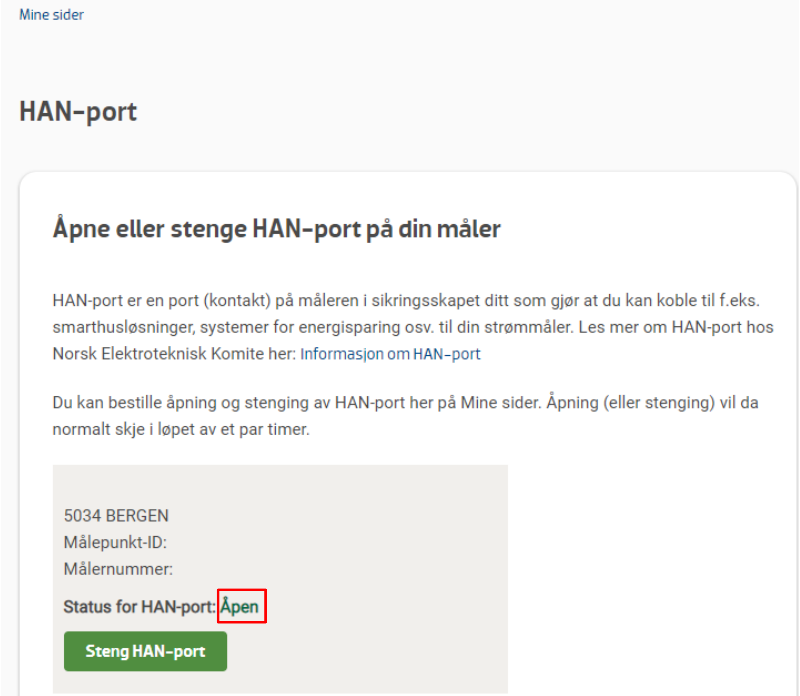
Åpne eller stenge HAN-port på din måler

5034 BERGEN
Målepunkt-ID:
Målernummer:

Status for HAN-port: Åpning bestilt (pågår)

✓ Åpning av HAN-port er bestilt. Normalt er bestilling fullført i løpet av et par timer. Dersom HAN-port ikke blir åpnet innen et par timer kan du kontakte vårt kundesenter for å få bistand.

Step 4: HAN-port has successfully opened



HAN-port

Åpne eller stenge HAN-port på din måler

HAN-port er en port (kontakt) på måleren i sikringsskapet ditt som gjør at du kan koble til f.eks. smarthusløsninger, systemer for energisparing osv. til din strømmåler. Les mer om HAN-port hos Norsk Elektroteknisk Komite her: Informasjon om HAN-port

Du kan bestille åpning og stenging av HAN-port her på Mine sider. Åpning (eller stenging) vil da normalt skje i løpet av et par timer.

5034 BERGEN
Målepunkt-ID:
Målernummer:

Status for HAN-port: Åpen

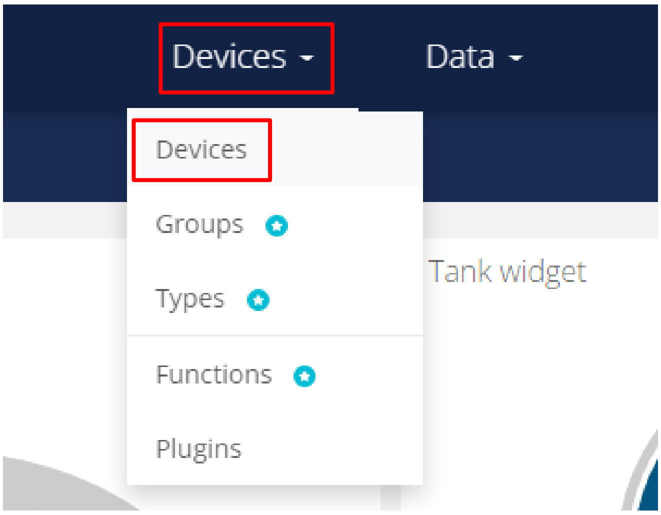
A.2 MQTT sign up

Step 1:

Create a **Ubidots** account

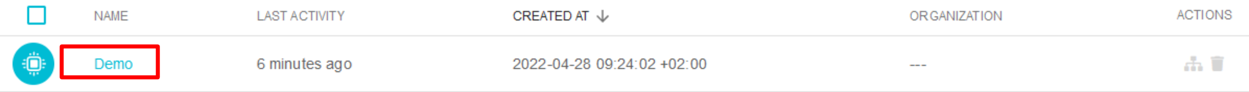
Step 2:

Log in to account and click the “**Devices**” → “**Devices**”



Step 3:

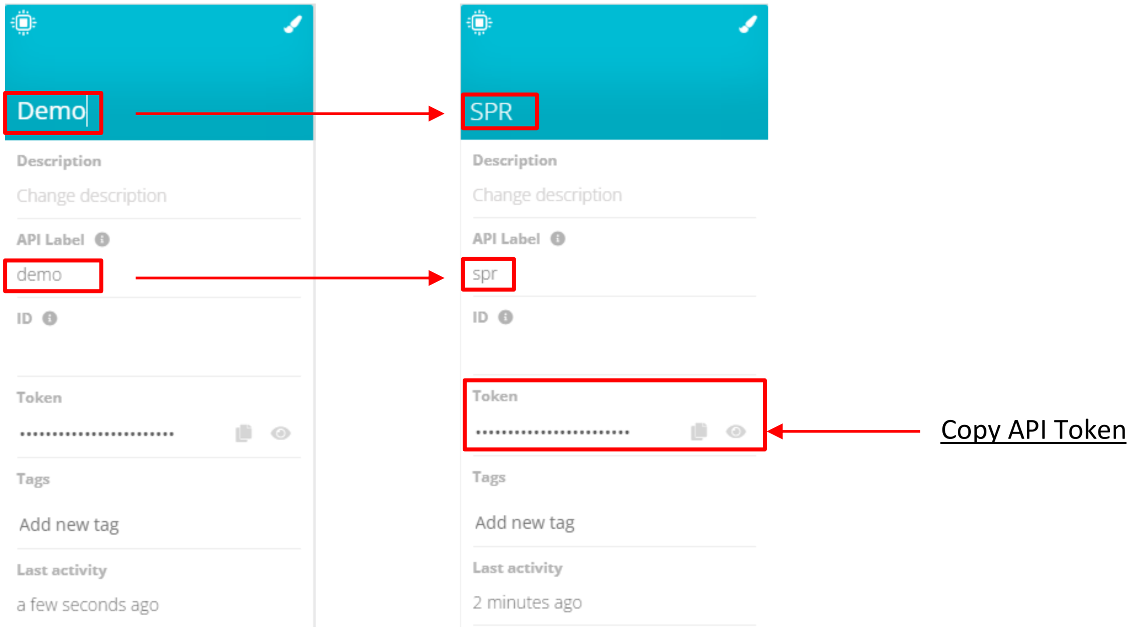
Click on “**Demo**” or a category that has been created



□	NAME	LAST ACTIVITY	CREATED AT ↓	ORGANIZATION	ACTIONS
	Demo	6 minutes ago	2022-04-28 09:24:02 +02:00	---	

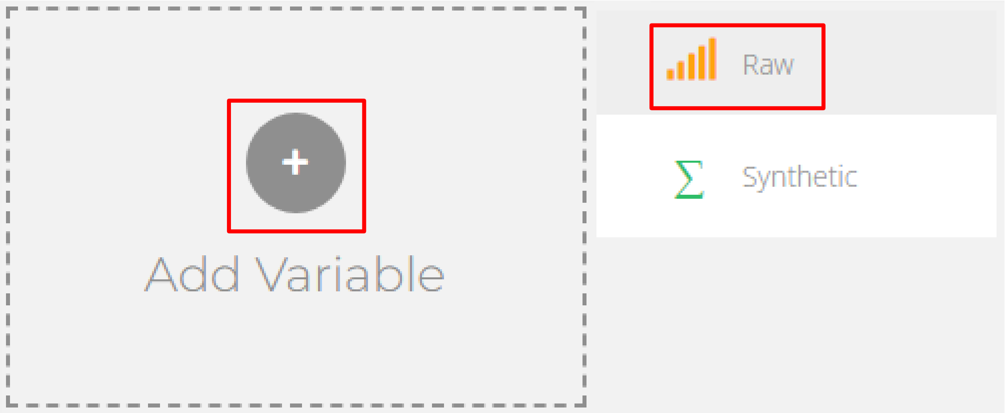
Step 4:

Change the name “**Demo**” to “**SPR**”, and copy the Token for later use



Step 5:

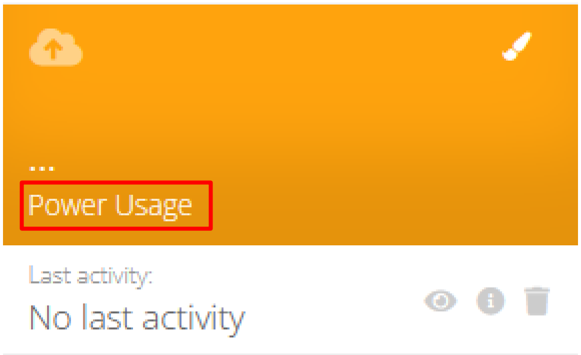
Add **2** variables by clicking the “**+**” → “**Raw**”



Step 6:

Change the name to both “**Power Usage**” and “**kWh**”

(NOTE: the name must be exact as mentioned)

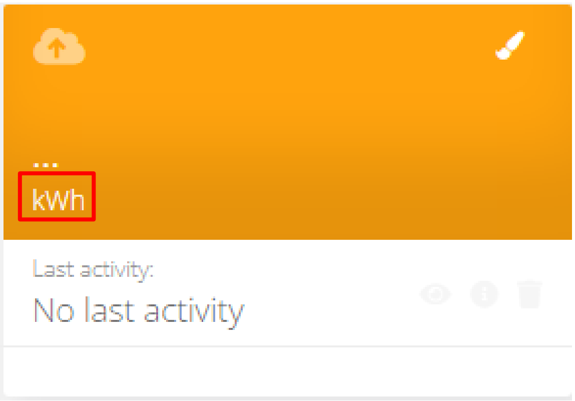


...

Power Usage

Last activity:

No last activity



...

kWh

Last activity:

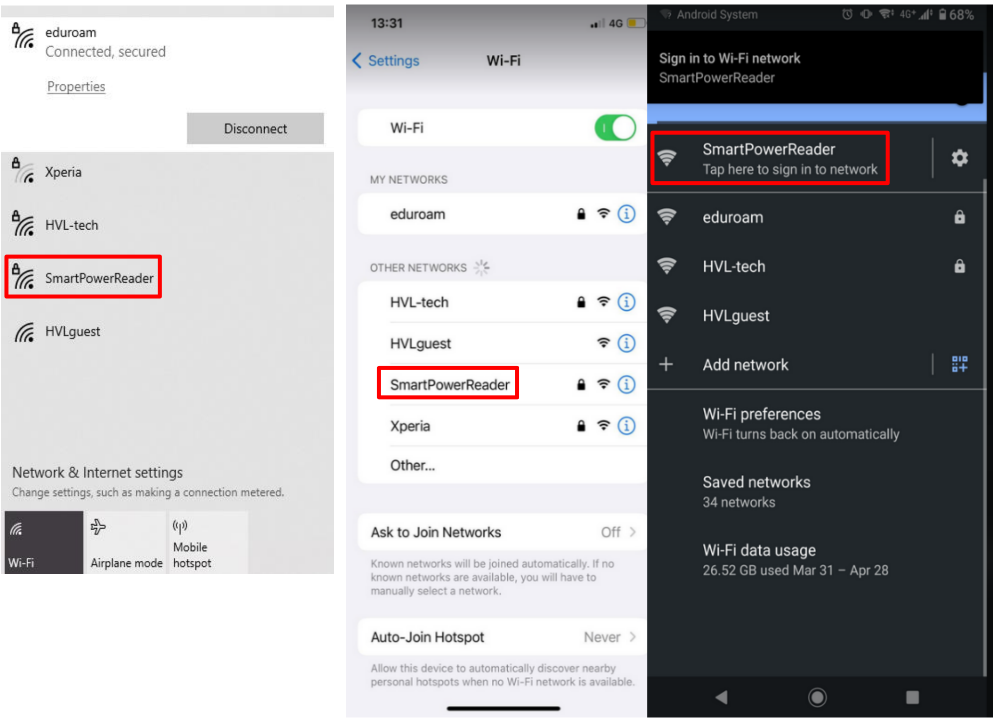
No last activity

A.3 Smart Power Reader

After creating an Ubidots account and **copying the Token**, and the Smart Power Reader device is **ON**, the user can then connect to the Access Point "**Smart Power Reader**". The Access Point can be found by using Wi-Fi

Step 1:

Connecting to Access Point "**Smart Power Reader**"



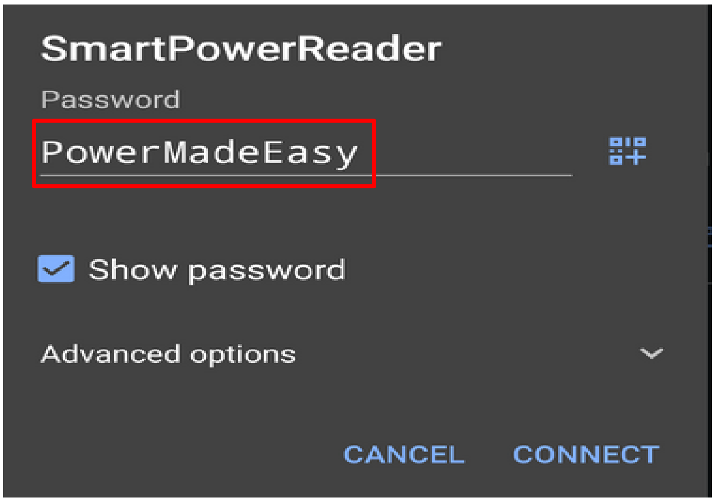
eduroam
Connected, secured
Properties
Disconnect
Xperia
HVL-tech
SmartPowerReader
HVLguest
Network & Internet settings
Change settings, such as making a connection metered.
Wi-Fi | Airplane mode | Mobile hotspot

13:31 4G
Settings Wi-Fi
Wi-Fi
MY NETWORKS
eduroam
OTHER NETWORKS
HVL-tech
HVLguest
SmartPowerReader
Xperia
Other...
Ask to Join Networks Off
Known networks will be joined automatically. If no known networks are available, you will have to manually select a network.
Auto-Join Hotspot Never
Allow this device to automatically discover nearby personal hotspots when no Wi-Fi network is available.

Android System 68%
Sign in to Wi-Fi network
SmartPowerReader
SmartPowerReader
Tap here to sign in to network
eduroam
HVL-tech
HVLguest
Add network
Wi-Fi preferences
Wi-Fi turns back on automatically
Saved networks
34 networks
Wi-Fi data usage
26.52 GB used Mar 31 – Apr 28

Step 2:

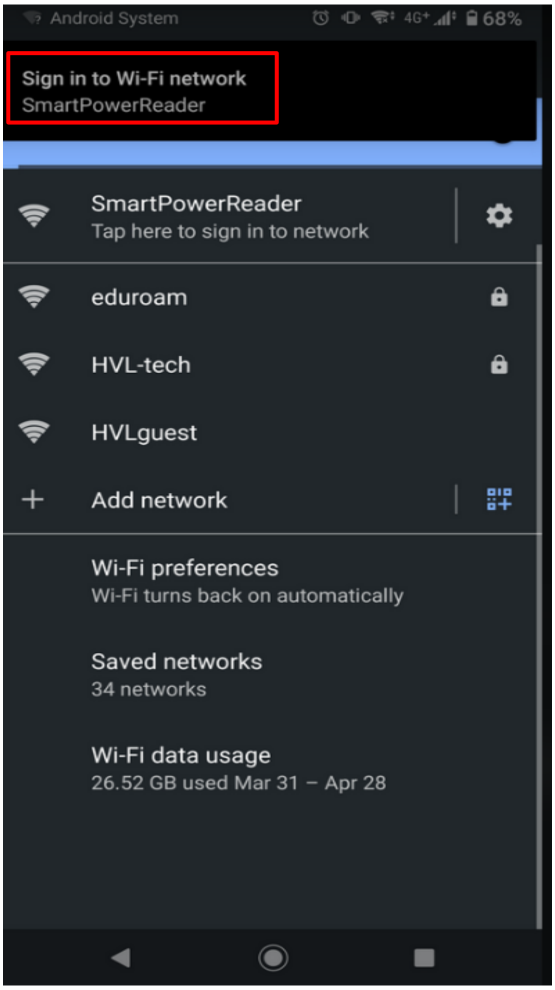
Default password: **PowerMadeEasy**



SmartPowerReader
Password
PowerMadeEasy
☑ Show password
Advanced options
CANCEL CONNECT

Step 3:

After successfully connecting to “**Smart Power Reader**”, the device should be led to a website **automatically**. If not, click on the “Smart Power Reader” or a pop-up message that states “**Sign in to Wi-Fi network**”



Android System 4G+ 68%

Sign in to Wi-Fi network
SmartPowerReader

SmartPowerReader
Tap here to sign in to network

eduroam

HVL-tech

HVLguest

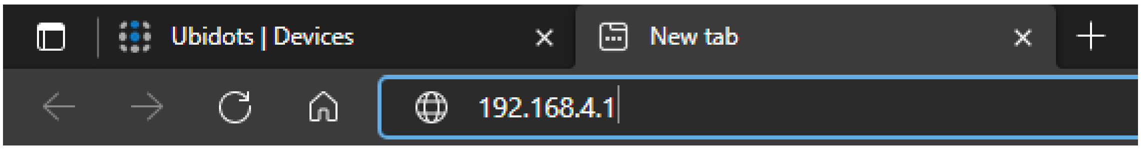
Add network

Wi-Fi preferences
Wi-Fi turns back on automatically

Saved networks
34 networks

Wi-Fi data usage
26.52 GB used Mar 31 – Apr 28

If the website does not pop-up by default, simply type in “**192.168.4.1**” on a link-field



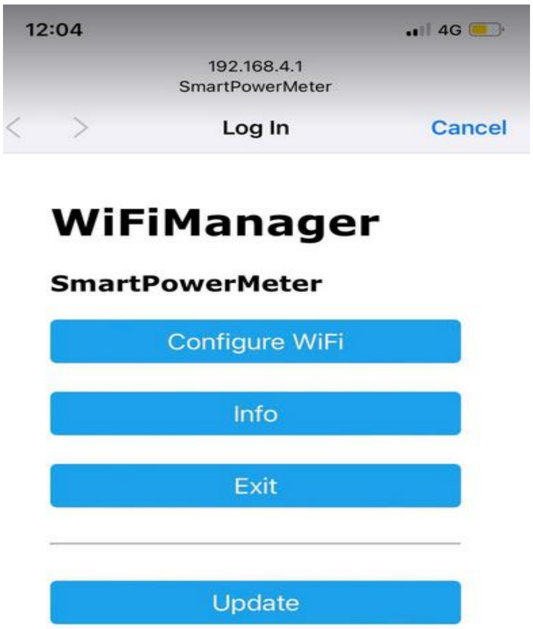
Ubidots | Devices

New tab

192.168.4.1

Step 4:

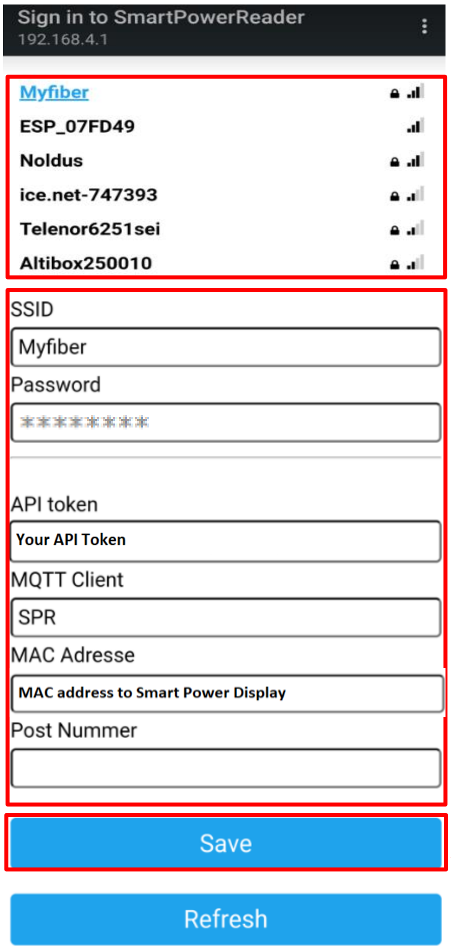
The website will have the following start page shown below



Step 5:

Click on the blue box "**Configure WiFi**" and wait for 3-5 seconds

Step 6:



Sign in to SmartPowerReader
192.168.4.1

- Myfiber
- ESP_07FD49
- Noldus
- ice.net-747393
- Telenor6251sei
- Altibox250010

SSID: Myfiber

Password: ********

API token: Your API Token

MQTT Client: SPR

MAC Adresse: MAC address to Smart Power Display

Post Nummer:

1. **Select Wi-Fi**
2. **Fill up the requirements**

   SSID:
   name of the Wi-Fi

   Password:
   Wi-Fi password

   API Token:
   copied from Ubidots

   MQTT Client:
   User define (default: SPR)

   MAC Adresse:
   On Smart Power Display show:
   10:97:BD:07:FD:48
   type in:
   1097bd07fd48 (Capital letters are not necessary)
3. **Save**

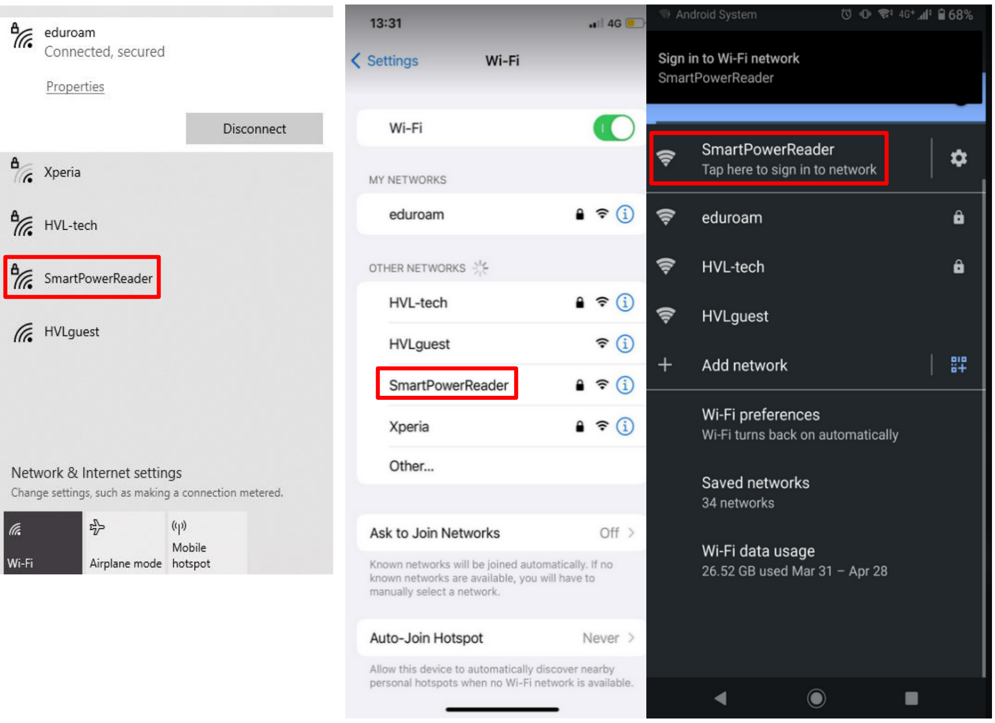
A.4 Smart Power Display

Step 1:

Start up the “Smart Power Display” using 5V Micro USB power supply. (picture)

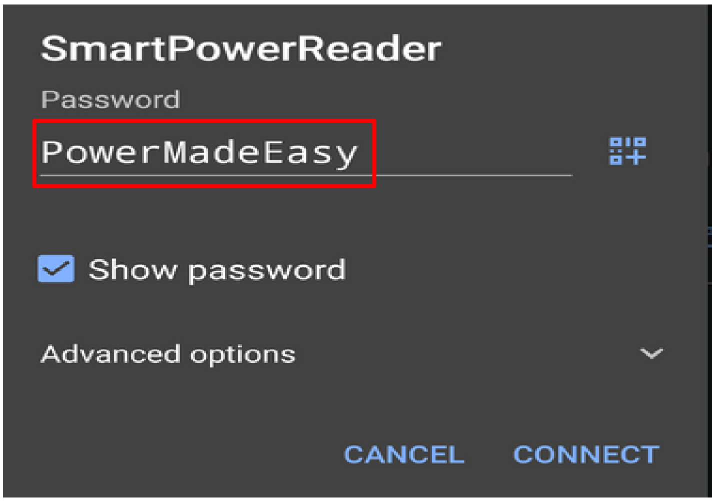
Step 2:

Connecting to Access Point “**Smart Power Display**”



Step 3:

Default password: **PowerMadeEasy**



SmartPowerReader

Password

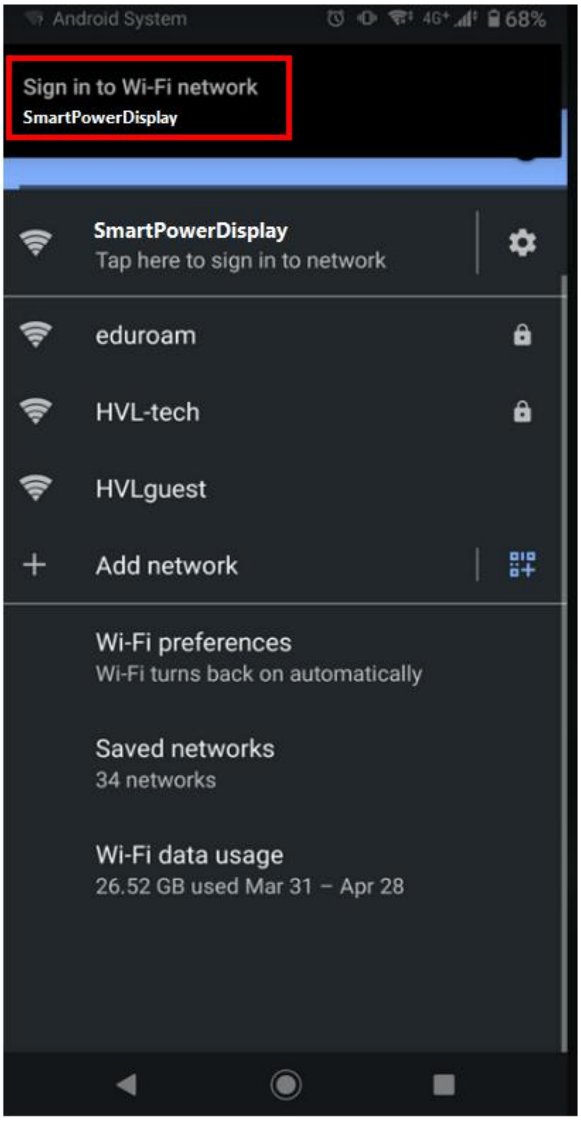
PowerMadeEasy

☑ Show password

Advanced options

Step 4:

After successfully connecting to “**Smart Power Display**”, the device should be led to a website **automatically**. If not, click on the “Smart Power Reader” or a pop-up message that states “Sign in to Wi-Fi network”



Android System 68%

Sign in to Wi-Fi network
SmartPowerDisplay

SmartPowerDisplay
Tap here to sign in to network

eduroam

HVL-tech

HVLguest

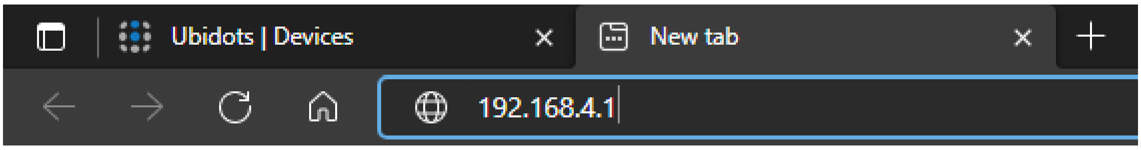
Add network

Wi-Fi preferences
Wi-Fi turns back on automatically

Saved networks
34 networks

Wi-Fi data usage
26.52 GB used Mar 31 – Apr 28

If the website does not pop-up by default, simply type in “**192.168.4.1**” on a link-field



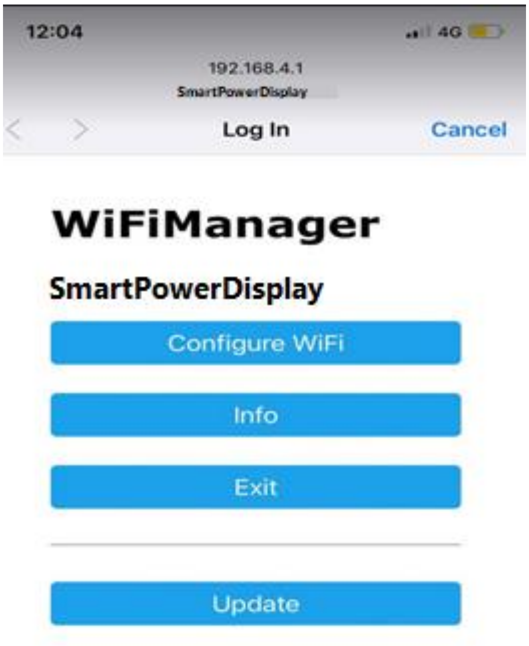
Ubidots | Devices

New tab

192.168.4.1

Step 5:

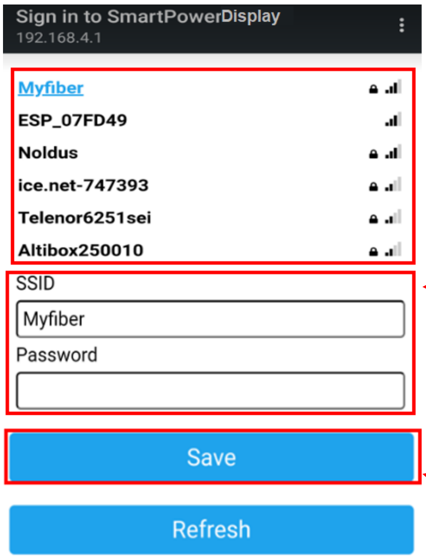
The website will have the following start page shown below



Step 6:

Click on the blue box “**Configure WiFi**” and wait for **3-5** seconds

Step 7:



SSID

Myfiber

Password

1. **Select Wi-Fi**
2. **Fill up the requirements**

   SSID:
   name of the Wi-Fi

   Password:
   Wi-Fi password
3. **Save**

A.5 SPR & SPD placement

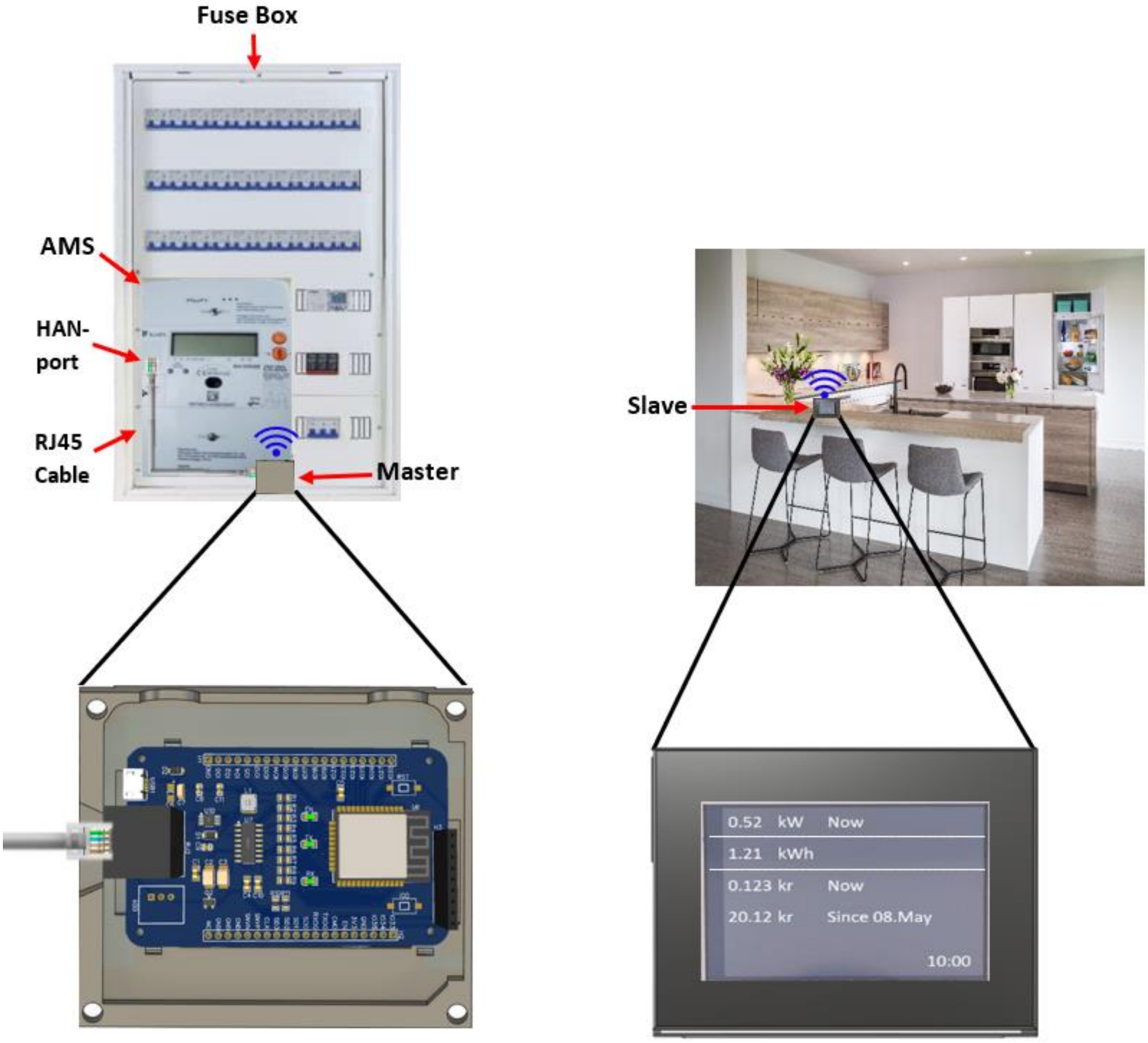
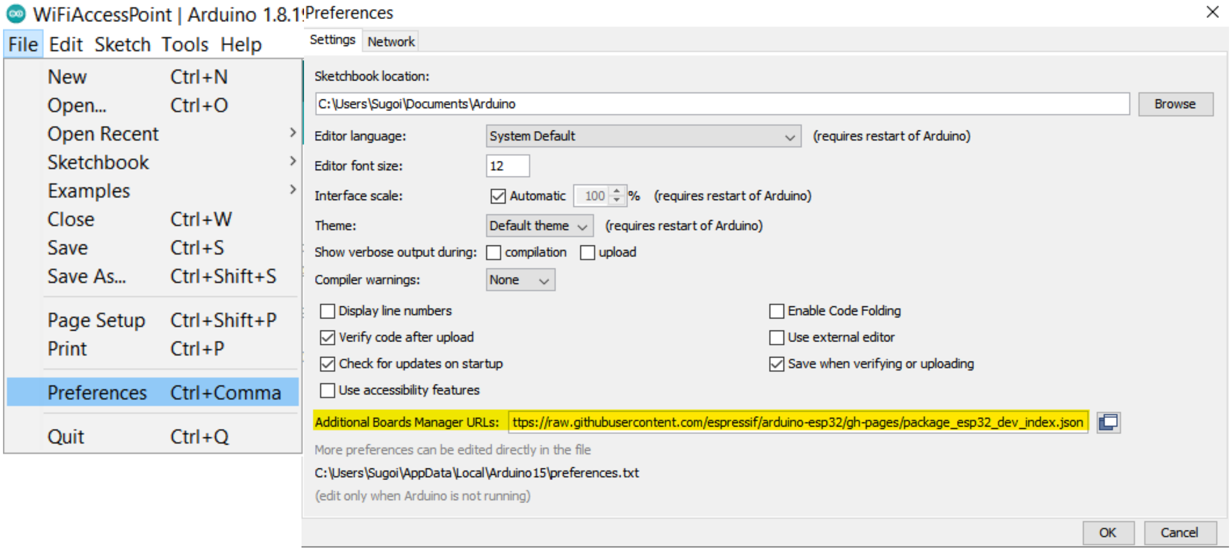


Figure 48 - General visualization of how the product works and where it is used

Appendix E. Setup Arduino IDE

Step 1: Install **Arduino IDE**

Step 2: Paste in ESP32 board manager **URL** in **Arduino → Tools → Preferences (shortcut: Ctrl + Comma)** : https://raw.githubusercontent.com/espressif/arduino-esp32/gh-pages/package_esp32_dev_index.json



Sketchbook location: C:\Users\Sugoi\Documents\Arduino

Editor language: System Default (requires restart of Arduino)

Editor font size: 12

Interface scale: ☑ Automatic 100 % (requires restart of Arduino)

Theme: Default theme (requires restart of Arduino)

Show verbose output during: □ compilation □ upload

Compiler warnings: None

□ Display line numbers

☑ Verify code after upload

☑ Check for updates on startup

□ Use accessibility features

□ Enable Code Folding

□ Use external editor

☑ Save when verifying or uploading

Additional Boards Manager URLs: ttps://raw.githubusercontent.com/espressif/arduino-esp32/gh-pages/package_esp32_dev_index.json

More preferences can be edited directly in the file

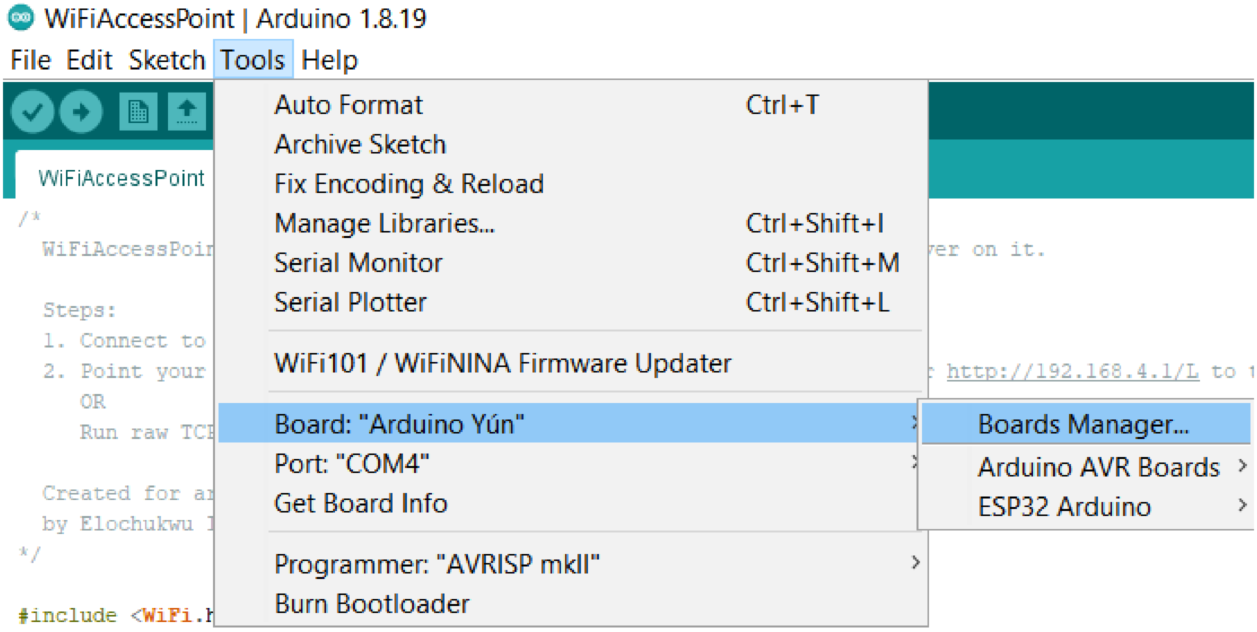
C:\Users\Sugoi\AppData\Local\Arduino15\preferences.txt

(edit only when Arduino is not running)

***Reference**:*

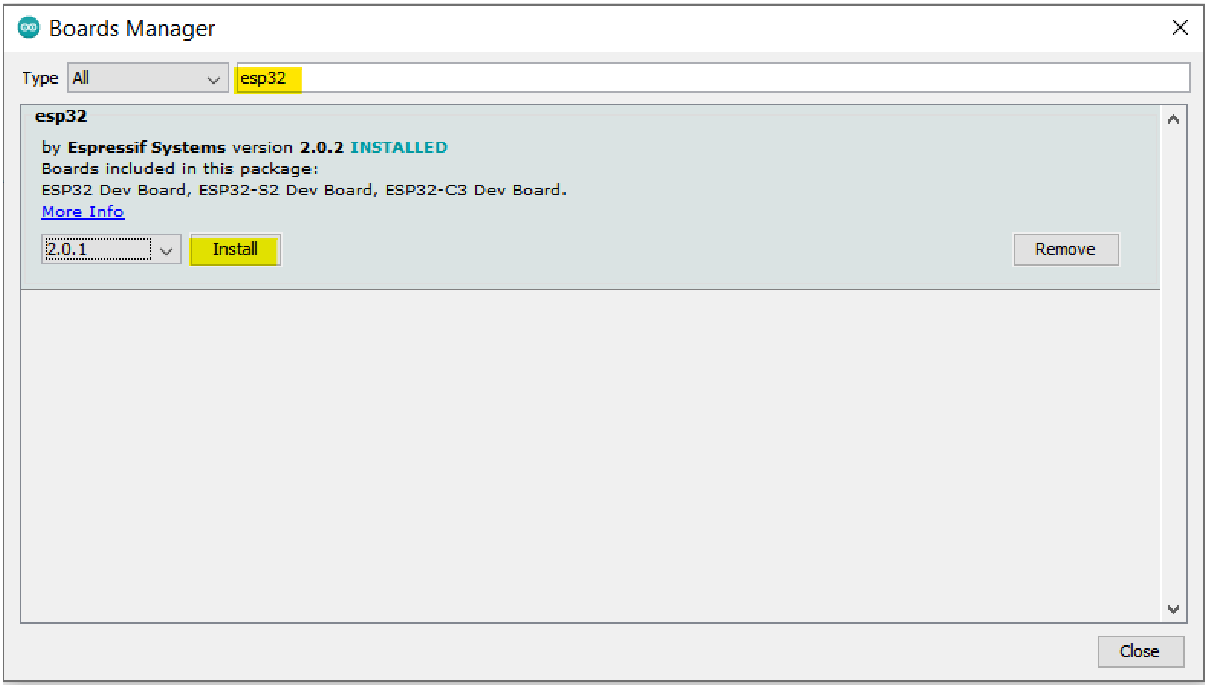
- https://randomnerdtutorials.com/installing-the-esp32-board-in-arduino-ide-windows-instructions/
- https://learn.adafruit.com/adafruit-metro-esp32-s2/arduino-ide-setup-2

Step 3: Open the Boards **Manager**. Go to **Tools** > **Board** > **Boards Manager…**


WiFiAccessPoint | Arduino 1.8.19
File Edit Sketch Tools Help
WiFiAccessPoint
Auto Format Ctrl+T
Archive Sketch
Fix Encoding & Reload
Manage Libraries... Ctrl+Shift+I
Serial Monitor Ctrl+Shift+M
Serial Plotter Ctrl+Shift+L
WiFi101 / WiFiNINA Firmware Updater
Board: "Arduino Yún"
Port: "COM4"
Get Board Info
Programmer: "AVRISP mkII"
Burn Bootloader
Boards Manager...
Arduino AVR Boards
ESP32 Arduino

```
/*
  WiFiAccessPoi                              er on it.

  Steps:
  1. Connect to
  2. Point your                              http://192.168.4.1/L to t
     OR
     Run raw TC

  Created for a
  by Elochukwu I
*/

#include <WiFi.h
```

Search for **ESP32** and install the “**ESP32 by Espressif Systems**”



Boards Manager

Type All esp32

esp32

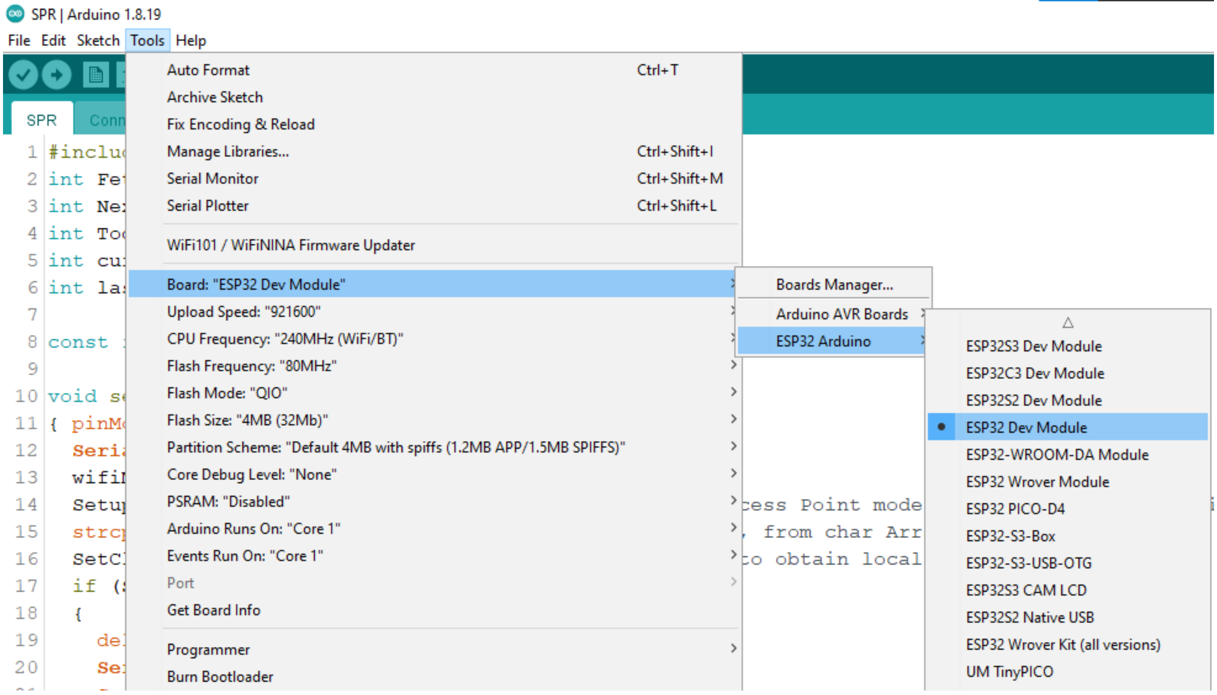
by **Espressif Systems** version **2.0.2** **INSTALLED**
Boards included in this package:
ESP32 Dev Board, ESP32-S2 Dev Board, ESP32-C3 Dev Board.
More Info

2.0.1 Install Remove

Close

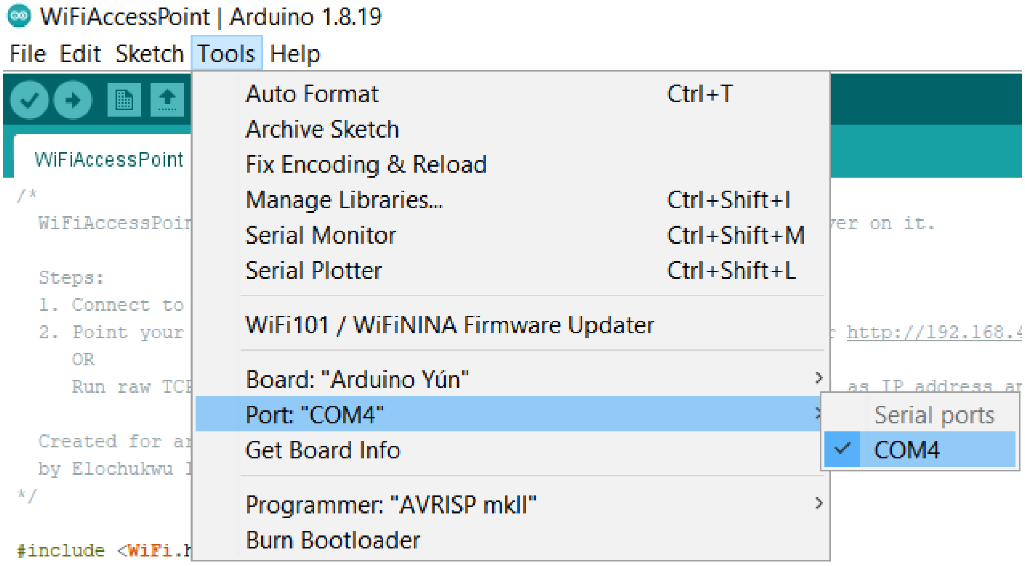
Testing the Installation

Step 1: Select your board in **Tools → Board** menu (in my case it's the **ESP32 Dev Module**)



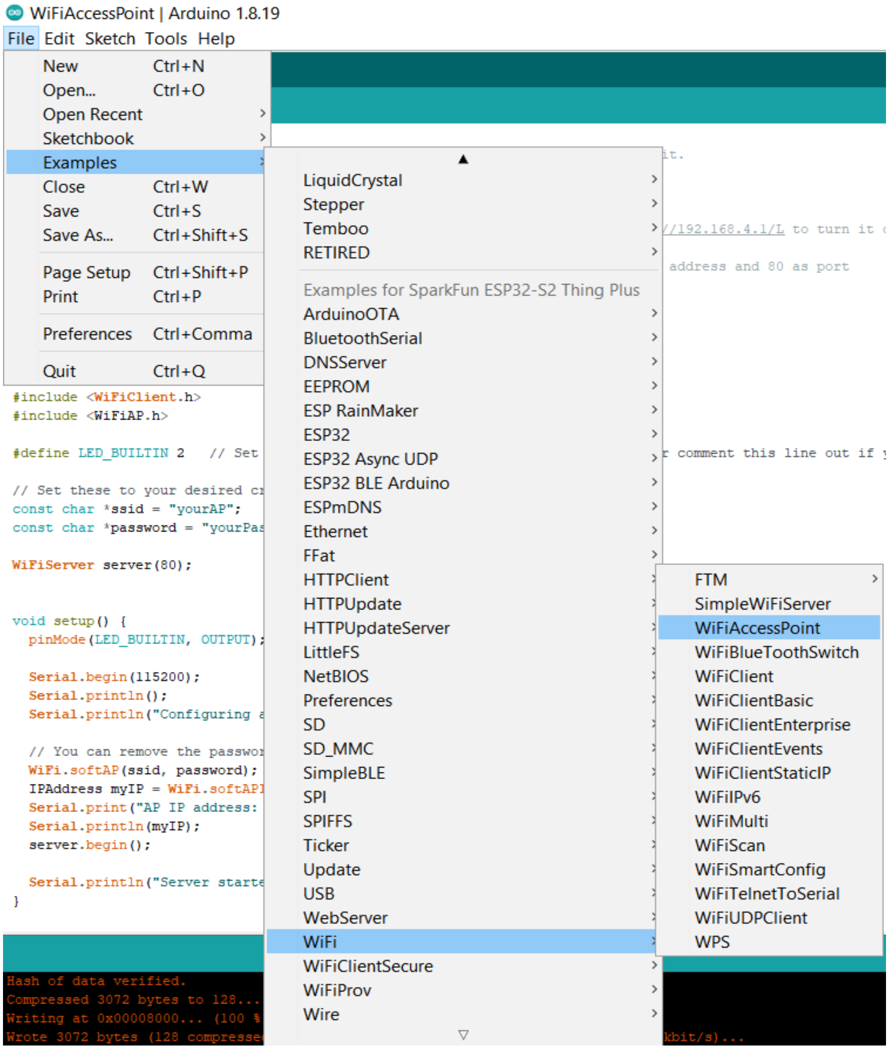
Board: "ESP32 Dev Module"
Upload Speed: "921600"
CPU Frequency: "240MHz (WiFi/BT)"
Flash Frequency: "80MHz"
Flash Mode: "QIO"
Flash Size: "4MB (32Mb)"
Partition Scheme: "Default 4MB with spiffs (1.2MB APP/1.5MB SPIFFS)"
Core Debug Level: "None"
PSRAM: "Disabled"
Arduino Runs On: "Core 1"
Events Run On: "Core 1"

Step 2: Select the Port (If there are several Ports to select, go to **Windows → Device Manager → Ports (Com & LPT)** then find the one that states **"CP210x USB to UART Bridge"**)


WiFiAccessPoint | Arduino 1.8.19
File Edit Sketch Tools Help
WiFiAccessPoint
Auto Format Ctrl+T
Archive Sketch
Fix Encoding & Reload
Manage Libraries... Ctrl+Shift+I
Serial Monitor Ctrl+Shift+M
Serial Plotter Ctrl+Shift+L
WiFi101 / WiFiNINA Firmware Updater
Get Board Info
Burn Bootloader
Serial ports
✓ COM4

Board: "Arduino Yún"
Port: "COM4"
Programmer: "AVRISP mkII"

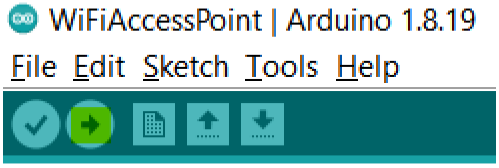
Step 3: Using example in **File → Examples → WiFi → WiFiAccessPoint**



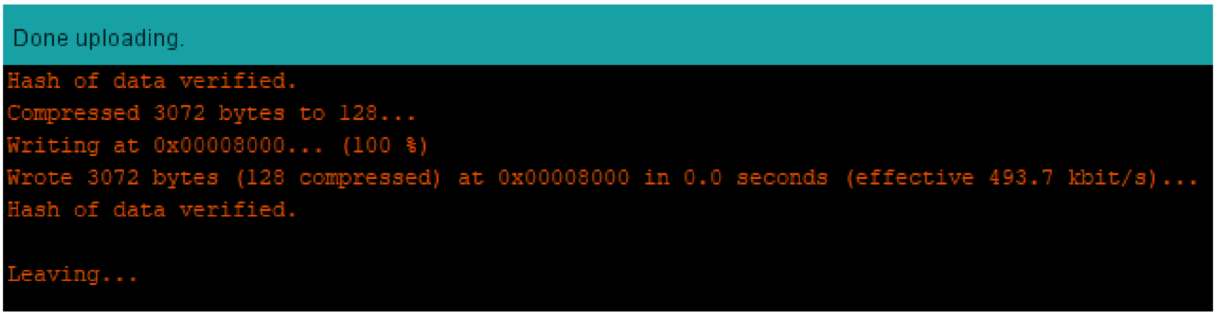
```
#include <WiFiClient.h>
#include <WiFiAP.h>

#define LED_BUILTIN 2   // Set                                  r comment this line out if y

// Set these to your desired cr
const char *ssid = "yourAP";
const char *password = "yourPas

WiFiServer server(80);


void setup() {
  pinMode(LED_BUILTIN, OUTPUT);

  Serial.begin(115200);
  Serial.println();
  Serial.println("Configuring a

  // You can remove the passwor
  WiFi.softAP(ssid, password);
  IPAddress myIP = WiFi.softAPI
  Serial.print("AP IP address:
  Serial.println(myIP);
  server.begin();

  Serial.println("Server starte
}
```

```
Hash of data verified.
Compressed 3072 bytes to 128...
Writing at 0x00008000... (100 %
Wrote 3072 bytes (128 compresse                                  kbit/s)...
```

Step 4: Upload the codes to ESP32 micro controller by clicking the arrow on the top left window in Arduino



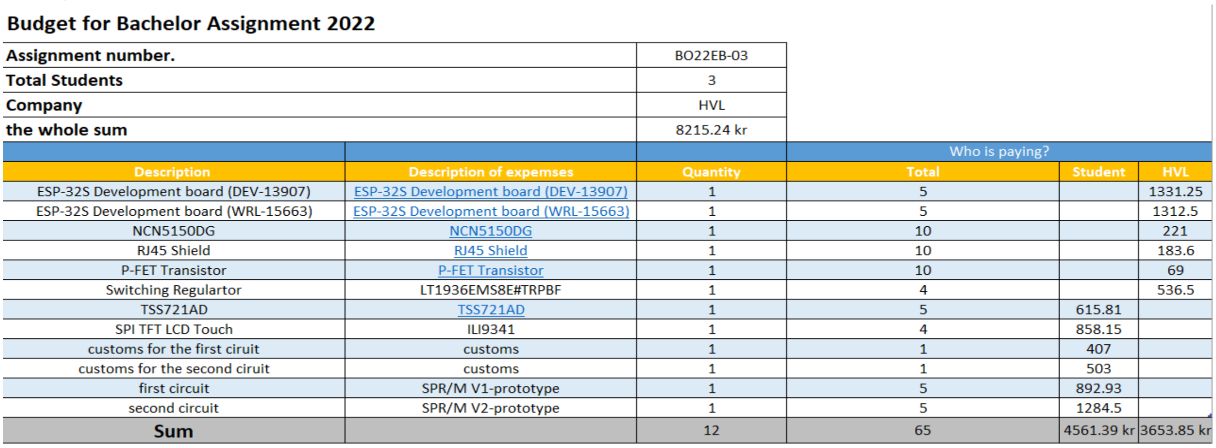
Once "**done uploading**", it would show the following message (not identical)



```
Done uploading.
Hash of data verified.
Compressed 3072 bytes to 128...
Writing at 0x00008000... (100 %)
Wrote 3072 bytes (128 compressed) at 0x00008000 in 0.0 seconds (effective 493.7 kbit/s)...
Hash of data verified.

Leaving...
```

Appendix F. Bill of materials

Budget list

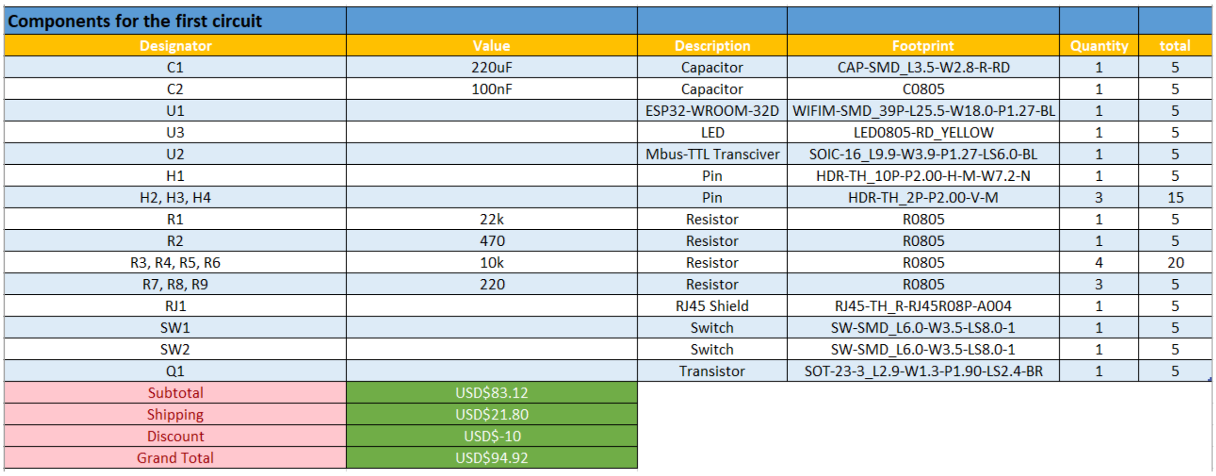


Budget for Bachelor Assignment 2022

Assignment number.	BO22EB-03
Total Students	3
Company	HVL
the whole sum	8215.24 kr

				Who is paying?	
Description	Description of expemses	Quantity	Total	Student	HVL
ESP-32S Development board (DEV-13907)	ESP-32S Development board (DEV-13907)	1	5		1331.25
ESP-32S Development board (WRL-15663)	ESP-32S Development board (WRL-15663)	1	5		1312.5
NCN5150DG	NCN5150DG	1	10		221
RJ45 Shield	RJ45 Shield	1	10		183.6
P-FET Transistor	P-FET Transistor	1	10		69
Switching Regulartor	LT1936EMS8E#TRPBF	1	4		536.5
TSS721AD	TSS721AD	1	5	615.81	
SPI TFT LCD Touch	ILI9341	1	4	858.15	
customs for the first ciruit	customs	1	1	407	
customs for the second ciruit	customs	1	1	503	
first circuit	SPR/M V1-prototype	1	5	892.93	
second circuit	SPR/M V2-prototype	1	5	1284.5	
Sum		12	65	4561.39 kr	3653.85 kr

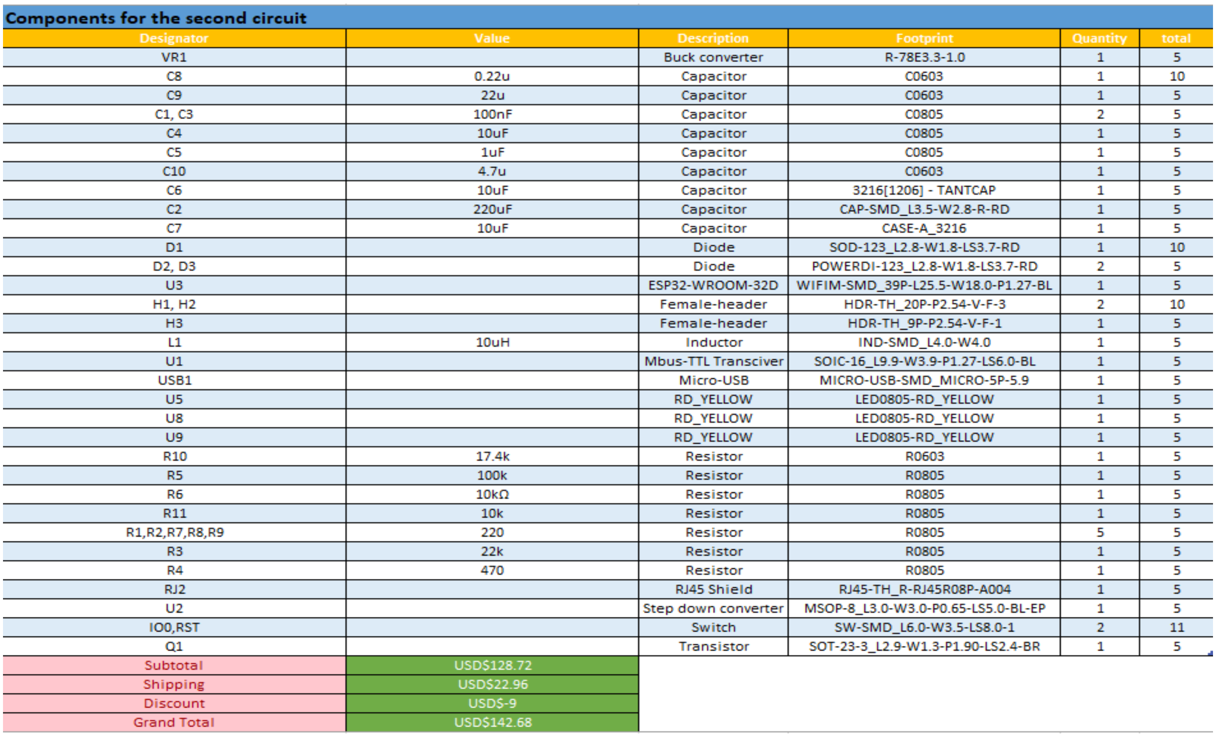
First circuit



Components for the first circuit

Designator	Value	Description	Footprint	Quantity	total
C1	220uF	Capacitor	CAP-SMD_L3.5-W2.8-R-RD	1	5
C2	100nF	Capacitor	C0805	1	5
U1		ESP32-WROOM-32D	WIFIM-SMD_39P-L25.5-W18.0-P1.27-BL	1	5
U3		LED	LED0805-RD_YELLOW	1	5
U2		Mbus-TTL Transciver	SOIC-16_L9.9-W3.9-P1.27-LS6.0-BL	1	5
H1		Pin	HDR-TH_10P-P2.00-H-M-W7.2-N	1	5
H2, H3, H4		Pin	HDR-TH_2P-P2.00-V-M	3	15
R1	22k	Resistor	R0805	1	5
R2	470	Resistor	R0805	1	5
R3, R4, R5, R6	10k	Resistor	R0805	4	20
R7, R8, R9	220	Resistor	R0805	3	5
RJ1		RJ45 Shield	RJ45-TH_R-RJ45R08P-A004	1	5
SW1		Switch	SW-SMD_L6.0-W3.5-LS8.0-1	1	5
SW2		Switch	SW-SMD_L6.0-W3.5-LS8.0-1	1	5
Q1		Transistor	SOT-23-3_L2.9-W1.3-P1.90-LS2.4-BR	1	5
Subtotal	USD$83.12				
Shipping	USD$21.80				
Discount	USD$-10				
Grand Total	USD$94.92				

Second circuit

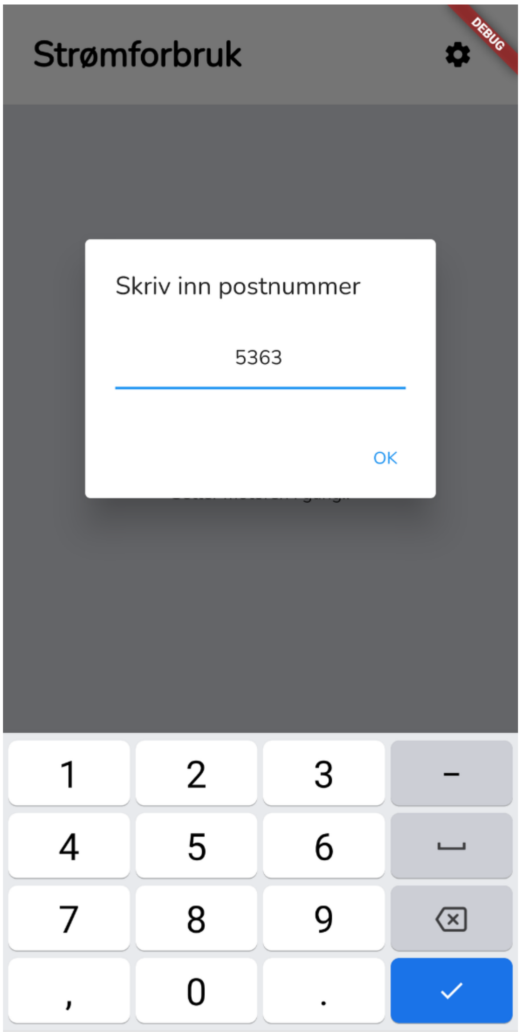


Components for the second circuit

Designator	Value	Description	Footprint	Quantity	total
VR1		Buck converter	R-78E3.3-1.0	1	5
C8	0.22u	Capacitor	C0603	1	10
C9	22u	Capacitor	C0603	1	5
C1, C3	100nF	Capacitor	C0805	2	5
C4	10uF	Capacitor	C0805	1	5
C5	1uF	Capacitor	C0805	1	5
C10	4.7u	Capacitor	C0603	1	5
C6	10uF	Capacitor	3216[1206] - TANTCAP	1	5
C2	220uF	Capacitor	CAP-SMD_L3.5-W2.8-R-RD	1	5
C7	10uF	Capacitor	CASE-A_3216	1	5
D1		Diode	SOD-123_L2.8-W1.8-LS3.7-RD	1	10
D2, D3		Diode	POWERDI-123_L2.8-W1.8-LS3.7-RD	2	5
U3		ESP32-WROOM-32D	WIFIM-SMD_39P-L25.5-W18.0-P1.27-BL	1	5
H1, H2		Female-header	HDR-TH_20P-P2.54-V-F-3	2	10
H3		Female-header	HDR-TH_9P-P2.54-V-F-1	1	5
L1	10uH	Inductor	IND-SMD_L4.0-W4.0	1	5
U1		Mbus-TTL Transciver	SOIC-16_L9.9-W3.9-P1.27-LS6.0-BL	1	5
USB1		Micro-USB	MICRO-USB-SMD_MICRO-5P-5.9	1	5
U5		RD_YELLOW	LED0805-RD_YELLOW	1	5
U8		RD_YELLOW	LED0805-RD_YELLOW	1	5
U9		RD_YELLOW	LED0805-RD_YELLOW	1	5
R10	17.4k	Resistor	R0603	1	5
R5	100k	Resistor	R0805	1	5
R6	10kΩ	Resistor	R0805	1	5
R11	10k	Resistor	R0805	1	5
R1,R2,R7,R8,R9	220	Resistor	R0805	5	5
R3	22k	Resistor	R0805	1	5
R4	470	Resistor	R0805	1	5
RJ2		RJ45 Shield	RJ45-TH_R-RJ45R08P-A004	1	5
U2		Step down converter	MSOP-8_L3.0-W3.0-P0.65-LS5.0-BL-EP	1	5
IO0,RST		Switch	SW-SMD_L6.0-W3.5-LS8.0-1	2	11
Q1		Transistor	SOT-23-3_L2.9-W1.3-P1.90-LS2.4-BR	1	5
Subtotal	USD$128.72				
Shipping	USD$22.96				
Discount	USD$-9				
Grand Total	USD$142.68				

Appendix G. Mobile Application

1. Postal code input

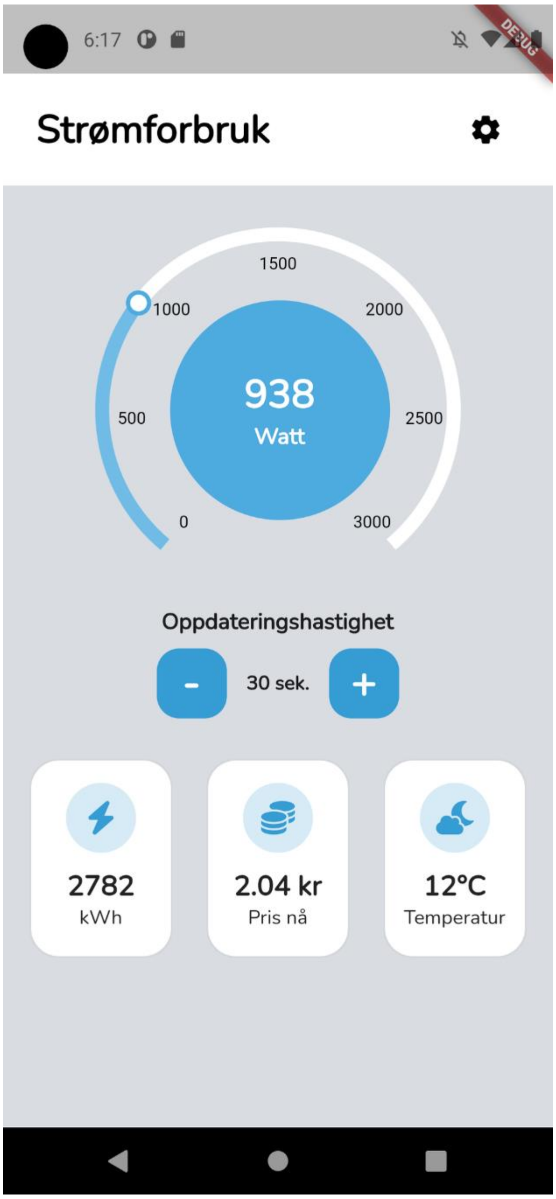


Skriv inn postnummer

5363

Figure 49 - Ability to adjust postal code

2. Power usage with a thirty second refresh rate



Strømforbruk

938 Watt

Oppdateringshastighet

30 sek.

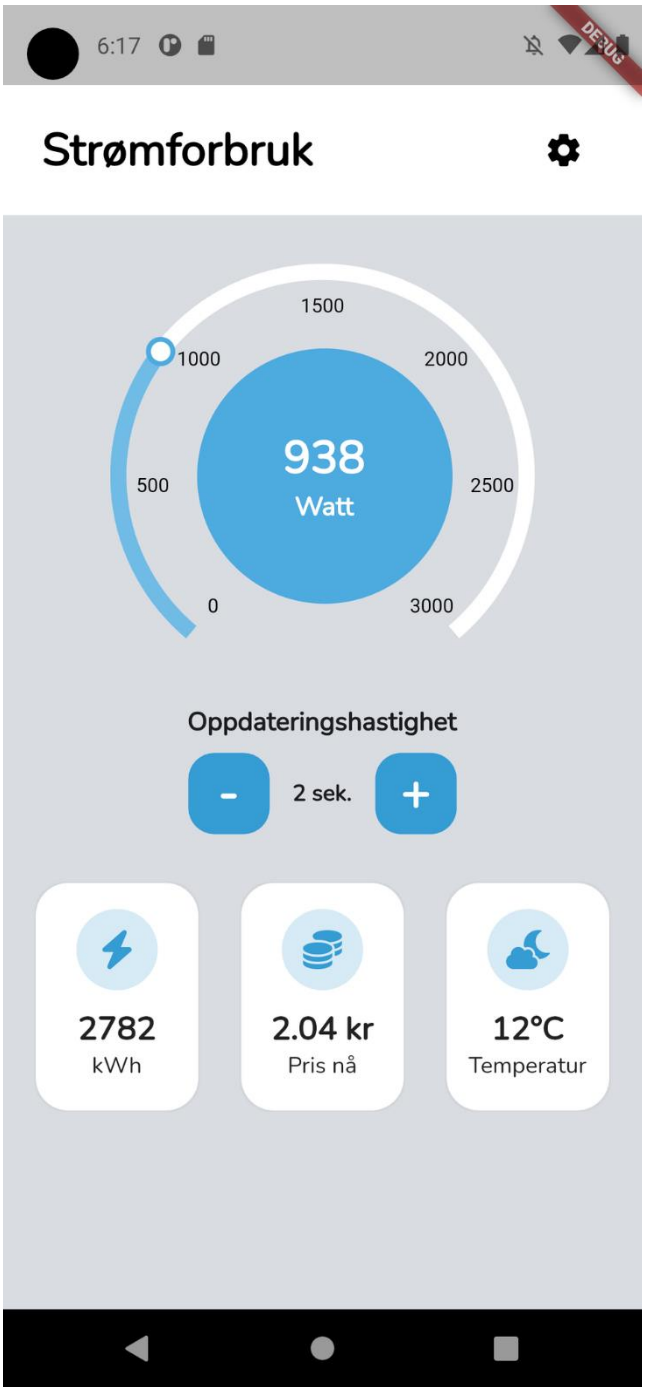
2782 kWh

2.04 kr Pris nå

12°C Temperatur

Figure 50 - User interface after setup

3. Power usage with a two second refresh rate



Strømforbruk

938 Watt

Oppdateringshastighet

2 sek.

2782 kWh

2.04 kr Pris nå

12°C Temperatur

Figure 51 - Refresh rate of 2 seconds

4. Spot price graph

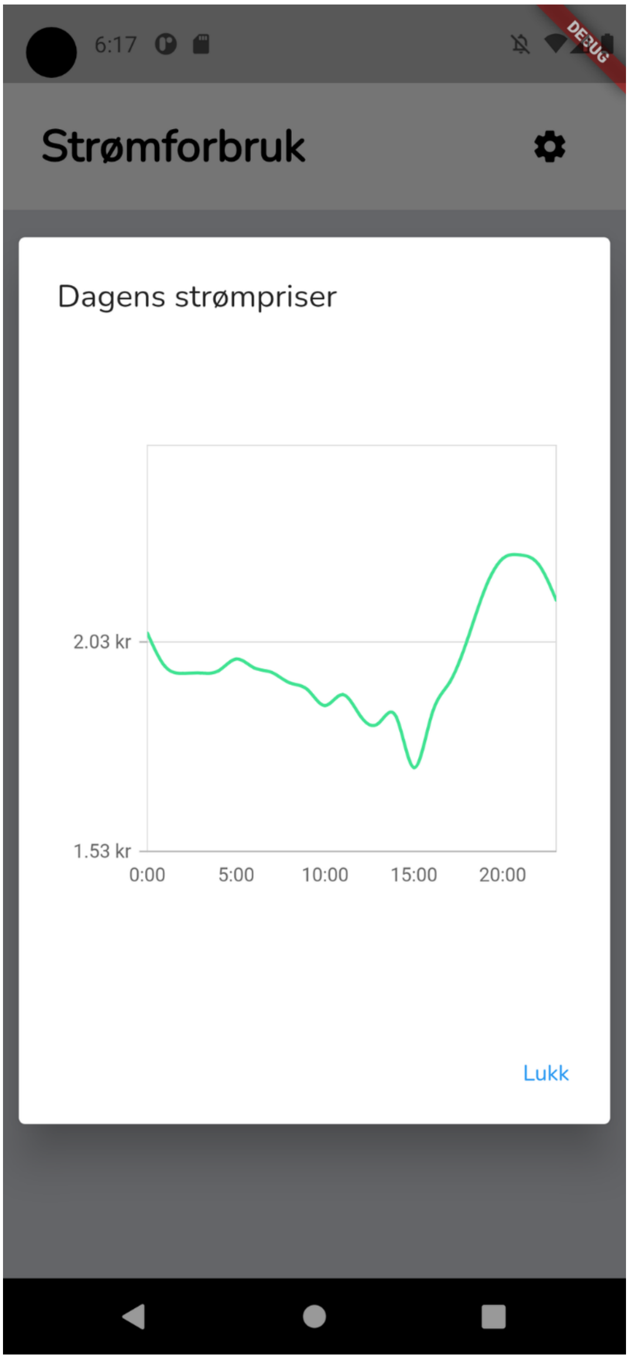
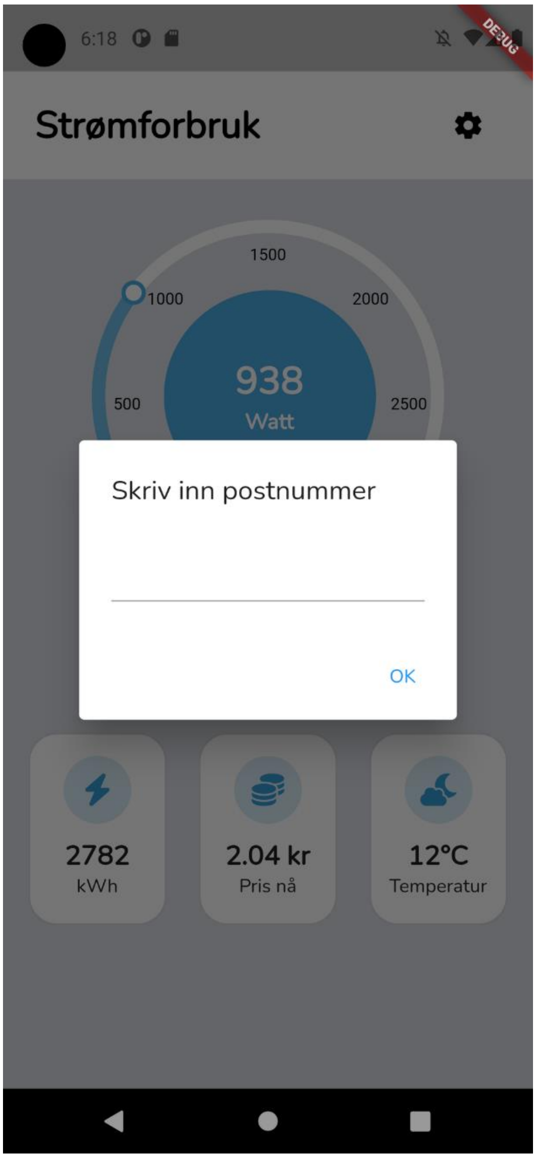


Figure 52 - Price distribution

5. Changing postal code after configuration



Strømforbruk

938
Watt

Skriv inn postnummer

2782
kWh

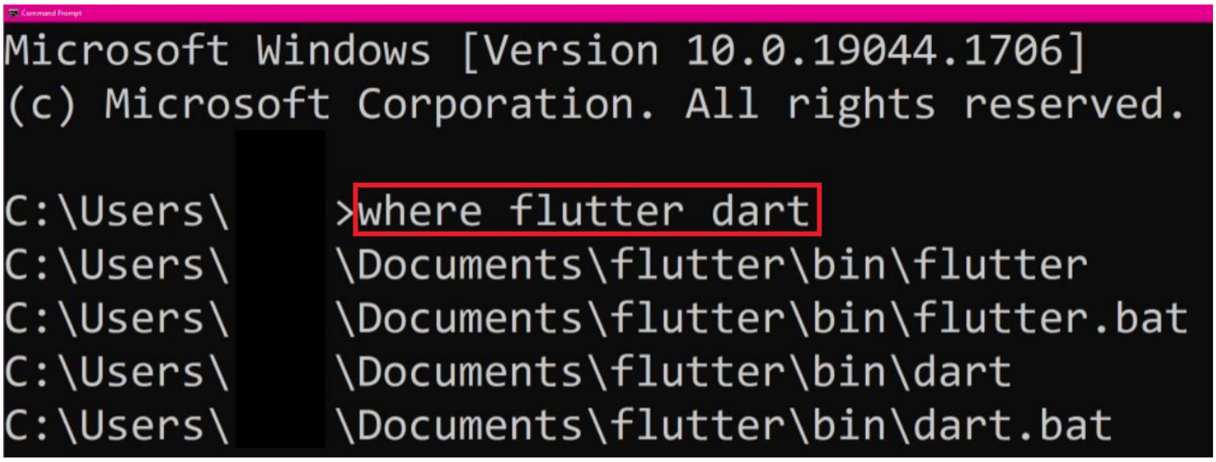
2.04 kr
Pris nå

12°C
Temperatur

Figure 53 - edit postal code

Appendix H. How to setup your PC for Flutter development

1. Head over to https://docs.flutter.dev/get-started/install/windows and download the zip file.
2. After the installation is completed, unzip the zip folder to a directory (e.g. C:\src\flutter)
3. Add flutter to your User Environment PATH variable by adding flutter\bin (C:\src\flutter\bin)
4. To check if flutter was added successfully – run the “where flutter dart” command in a new console window

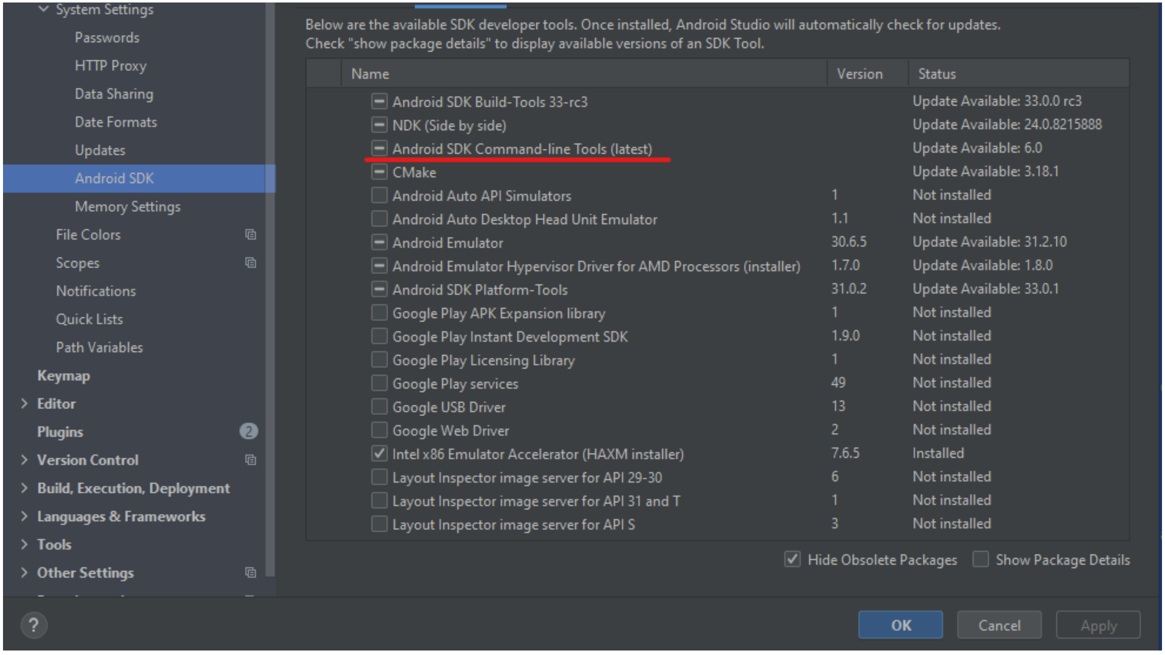


```
Microsoft Windows [Version 10.0.19044.1706]
(c) Microsoft Corporation. All rights reserved.

C:\Users\[redacted]>where flutter dart
C:\Users\    \Documents\flutter\bin\flutter
C:\Users\    \Documents\flutter\bin\flutter.bat
C:\Users\    \Documents\flutter\bin\dart
C:\Users\    \Documents\flutter\bin\dart.bat
```

5. Run flutter doctor to check which steps are remaining to complete the Flutter installation.

Appendix I. Setup Android Studio and Android SDK

1. Download Android Studio from https://developer.android.com/studio
2. Agree Android Licenses by running “flutter doctor --android-licenses” command
3. Enabling Android SDK Command-line Tools



Below are the available SDK developer tools. Once installed, Android Studio will automatically check for updates.
Check "show package details" to display available versions of an SDK Tool.

Name	Version	Status
⊟ Android SDK Build-Tools 33-rc3		Update Available: 33.0.0 rc3
⊟ NDK (Side by side)		Update Available: 24.0.8215888
⊟ Android SDK Command-line Tools (latest)		Update Available: 6.0
⊟ CMake		Update Available: 3.18.1
☐ Android Auto API Simulators	1	Not installed
☐ Android Auto Desktop Head Unit Emulator	1.1	Not installed
⊟ Android Emulator	30.6.5	Update Available: 31.2.10
⊟ Android Emulator Hypervisor Driver for AMD Processors (installer)	1.7.0	Update Available: 1.8.0
⊟ Android SDK Platform-Tools	31.0.2	Update Available: 33.0.1
☐ Google Play APK Expansion library	1	Not installed
☐ Google Play Instant Development SDK	1.9.0	Not installed
☐ Google Play Licensing Library	1	Not installed
☐ Google Play services	49	Not installed
☐ Google USB Driver	13	Not installed
☐ Google Web Driver	2	Not installed
☑ Intel x86 Emulator Accelerator (HAXM installer)	7.6.5	Installed
☐ Layout Inspector image server for API 29-30	6	Not installed
☐ Layout Inspector image server for API 31 and T	1	Not installed
☐ Layout Inspector image server for API S	3	Not installed

☑ Hide Obsolete Packages ☐ Show Package Details

Appendix J. Testing circuits

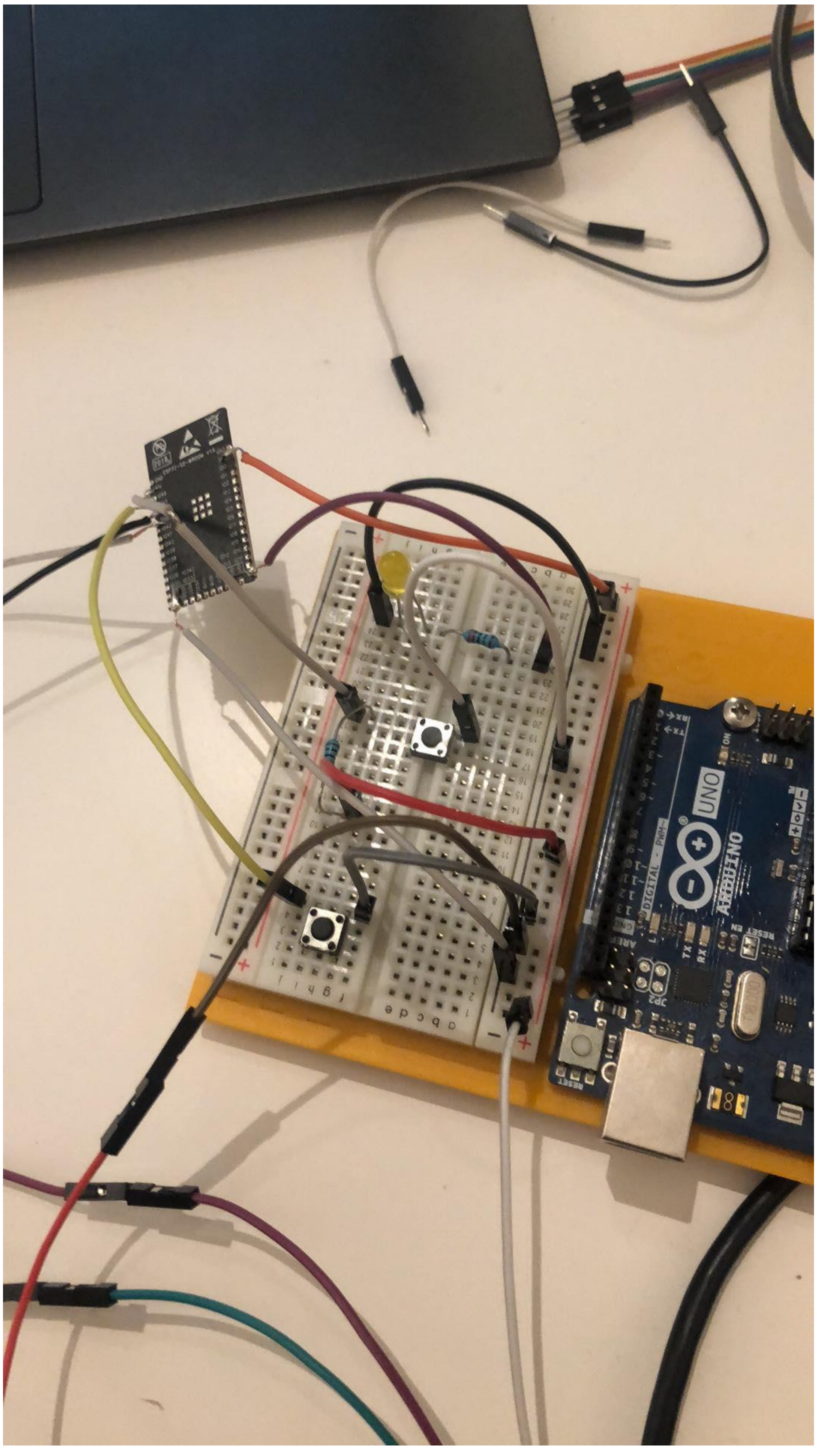


Figure 54 - testing desoldered ESP32

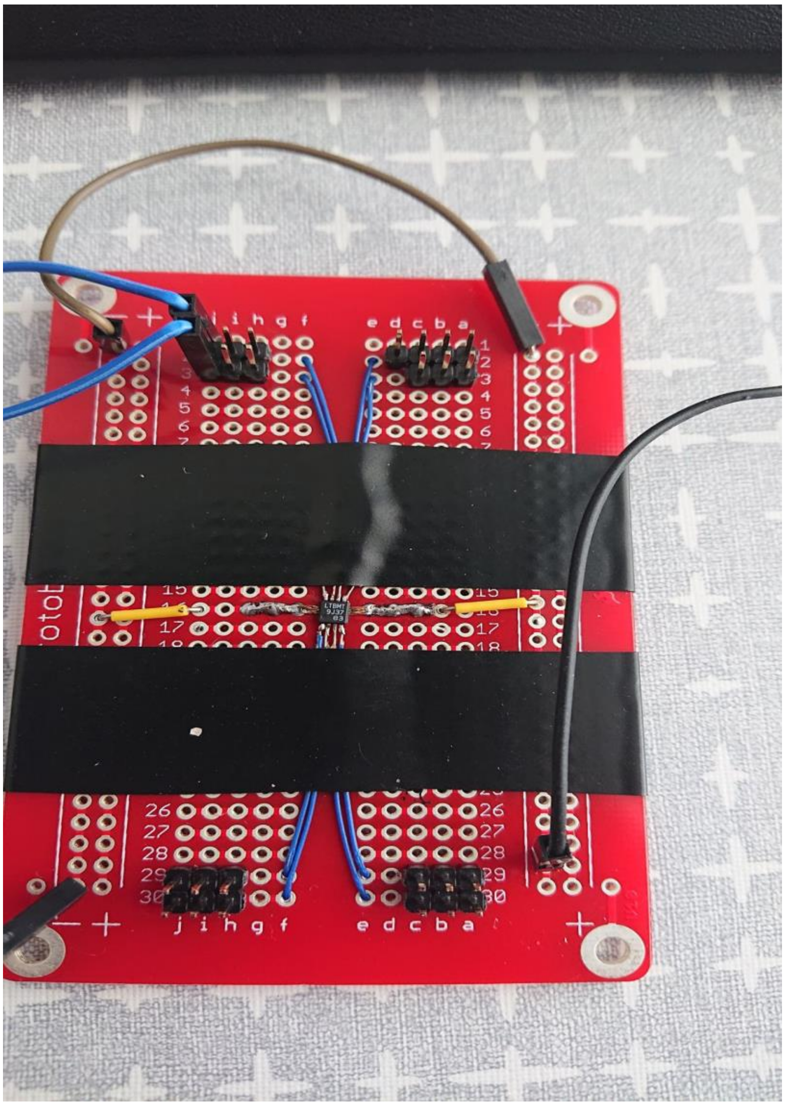


Figure 55 - testing LT1936

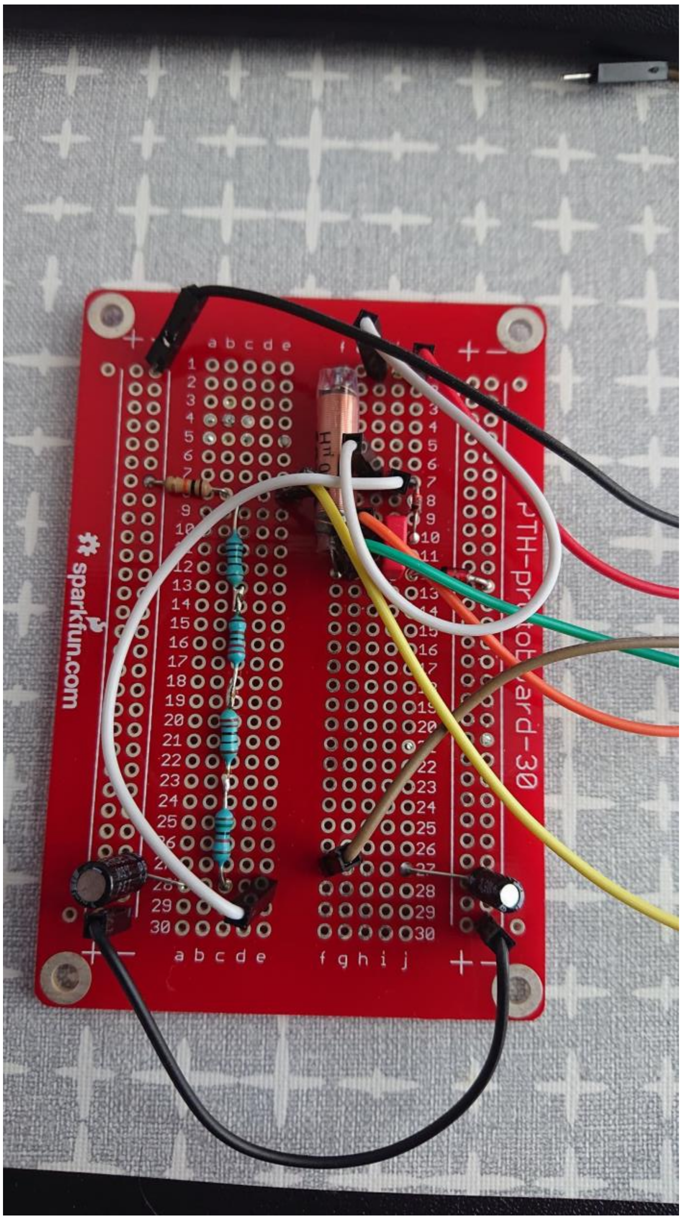
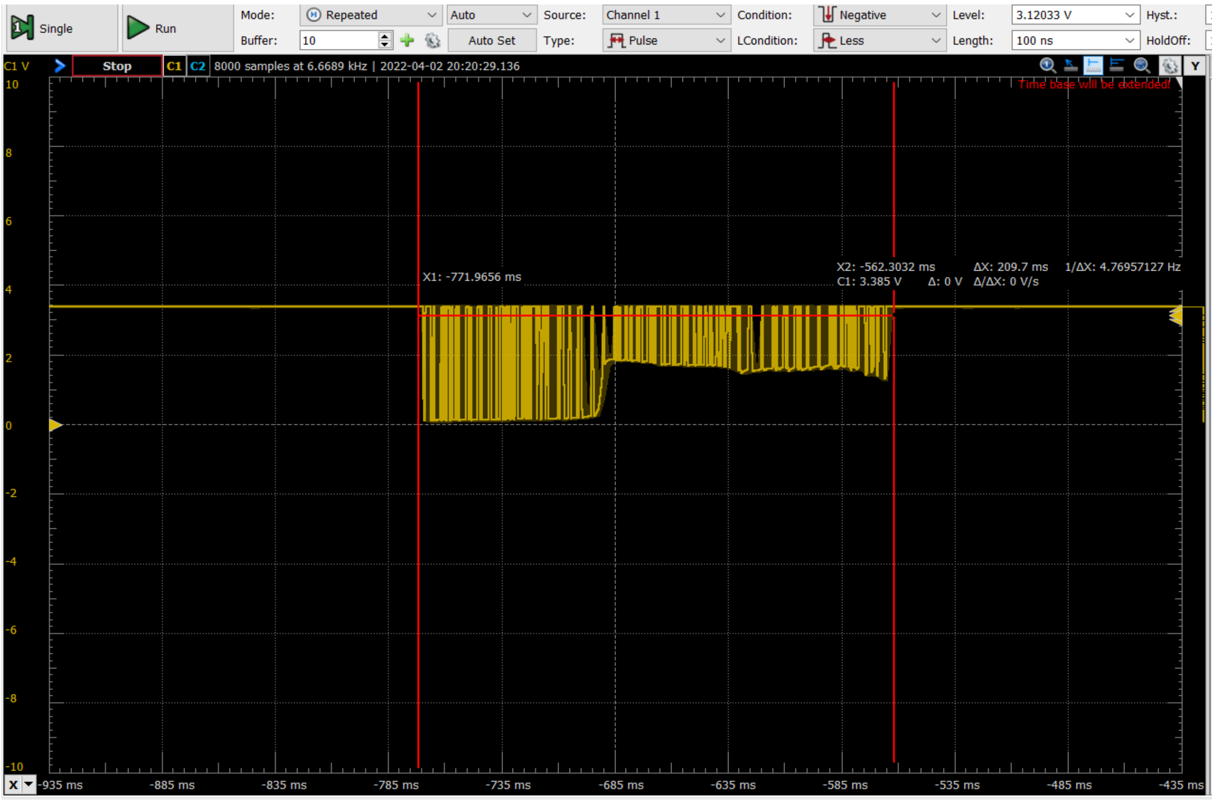


Figure 56 - Testing circuit for LT1936

Appendix K. List type

Small package (2 seconds) received using oscilloscope software



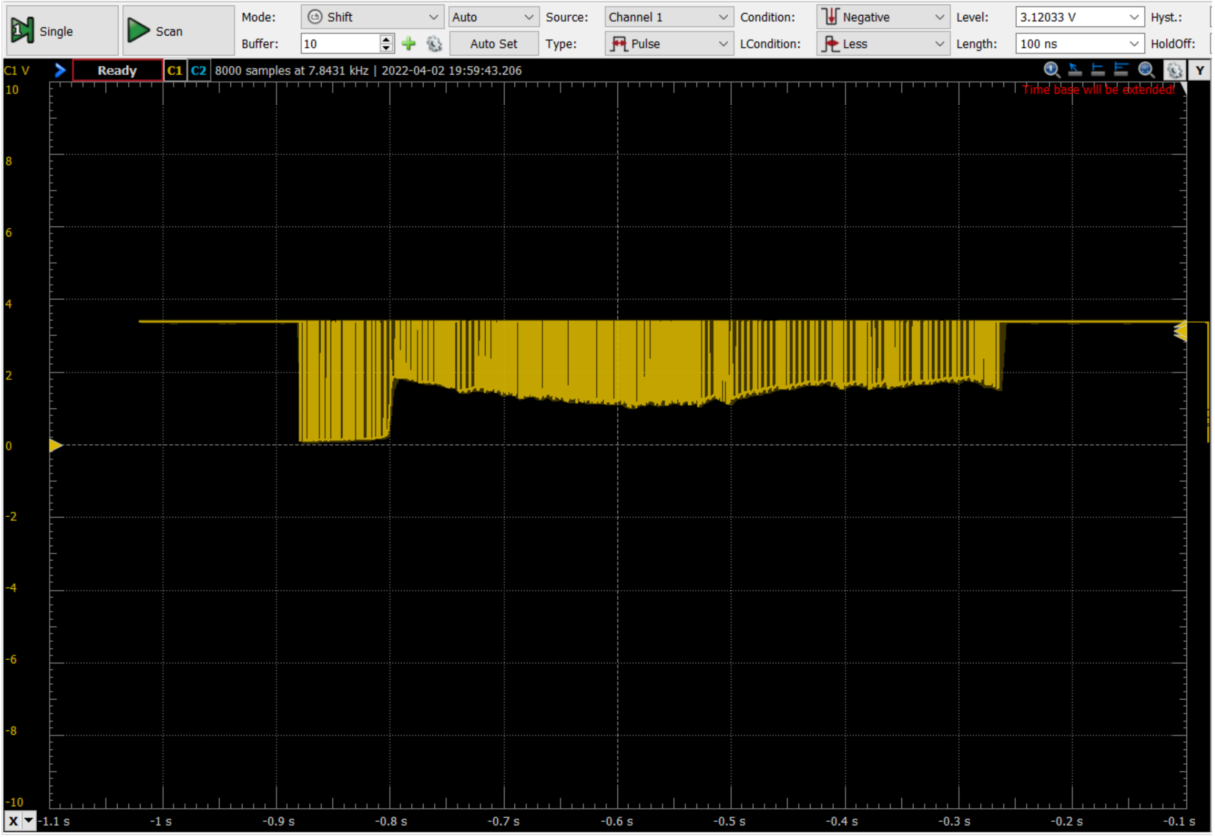
Mode: Repeated
Buffer: 10
Auto
Auto Set
Source: Channel 1
Type: Pulse
Condition: Negative
LCondition: Less
Level: 3.12033 V
Length: 100 ns
Hyst.:
HoldOff:

Stop | 8000 samples at 6.6689 kHz | 2022-04-02 20:20:29.136

Time base will be extended!

X1: -771.9656 ms
X2: -562.3032 ms
C1: 3.385 V
ΔX: 209.7 ms
Δ: 0 V
1/ΔX: 4.76957127 Hz
Δ/ΔX: 0 V/s

Medium package (10 seconds) received using oscilloscope software

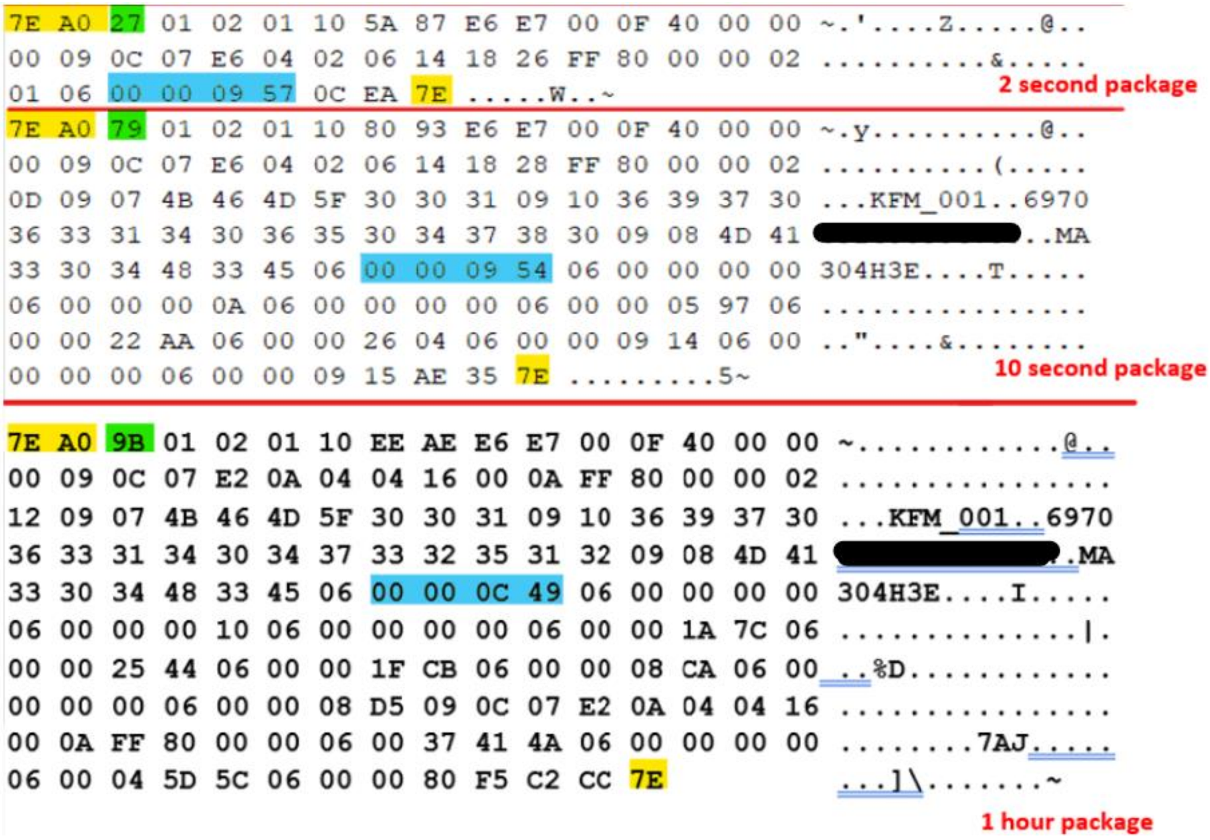


Mode: Shift
Buffer: 10
Auto
Auto Set
Source: Channel 1
Type: Pulse
Condition: Negative
LCondition: Less
Level: 3.12033 V
Length: 100 ns
Hyst.:
HoldOff:

Ready | 8000 samples at 7.8431 kHz | 2022-04-02 19:59:43.206

Time base will be extended!

Package of data shown in **Hexadecimal** form received by serial monitor in Arduino with **2400** baud rate



```
7E A0 27 01 02 01 10 5A 87 E6 E7 00 0F 40 00 00 ~.'....Z.....@..
00 09 0C 07 E6 04 02 06 14 18 26 FF 80 00 00 02 ..........&.....
01 06 00 00 09 57 0C EA 7E .....W..~
7E A0 79 01 02 01 10 80 93 E6 E7 00 0F 40 00 00 ~.y..........@..
00 09 0C 07 E6 04 02 06 14 18 28 FF 80 00 00 02 ..........(.....
0D 09 07 4B 46 4D 5F 30 30 31 09 10 36 39 37 30 ...KFM_001..6970
36 33 31 34 30 36 35 30 34 37 38 30 09 08 4D 41 [redacted]..MA
33 30 34 48 33 45 06 00 00 09 54 06 00 00 00 00 304H3E....T.....
06 00 00 00 0A 06 00 00 00 00 06 00 00 05 97 06 ................
00 00 22 AA 06 00 00 26 04 06 00 00 09 14 06 00 .."....&........
00 00 00 06 00 00 09 15 AE 35 7E .........5~

7E A0 9B 01 02 01 10 EE AE E6 E7 00 0F 40 00 00 ~.............@..
00 09 0C 07 E2 0A 04 04 16 00 0A FF 80 00 00 02 ................
12 09 07 4B 46 4D 5F 30 30 31 09 10 36 39 37 30 ...KFM_001..6970
36 33 31 34 30 34 37 33 32 35 31 32 09 08 4D 41 [redacted].MA
33 30 34 48 33 45 06 00 00 0C 49 06 00 00 00 00 304H3E....I.....
06 00 00 00 10 06 00 00 00 00 06 00 00 1A 7C 06 ..............|.
00 00 25 44 06 00 00 1F CB 06 00 00 08 CA 06 00 ..%D............
00 00 00 06 00 00 08 D5 09 0C 07 E2 0A 04 04 16 ................
00 0A FF 80 00 00 06 00 37 41 4A 06 00 00 00 00 ........7AJ.....
06 00 04 5D 5C 06 00 00 80 F5 C2 CC 7E ...]\.......~
```

2 second package

10 second package

1 hour package

The picture above shows the structure of a package which always starts with “7E A0” and ends with “7E” in **Hexadecimal**.

Start/end bytes

Package type

Effect

The key information we need from the package is the start bytes (marked in Yellow) and the package type (marked in Light blue). With those two combined we can then extract other information we need for our project. Our project focuses on one’s power usage in a given the moment and total usage by the end of the day which we can simply extract the information from the package we receive according to its type. Once the package of power usage has been successfully extracted, we can then send it to both the MQTT cloud service and another device(s).

Data form example: Smart meter part 1: Getting the meter data - Kode24 [38]

Appendix L. Arduino libraries

To write a code that allows for completion of the project, picking the right libraries is essential. Underneath is a list of libraries that were used and a short description of their role in this project.

WiFi.h	Allows the ESP32 to connect to Wi-Fi using the SSID and password to connect
HTTPClient.h	Used to fetch API data through a URI link.
ArduinoJson.h	Used to serialize and deserialize JSON. Very useful for fetching and saving data
esp_now.h	Necessary for ESP-NOW connection between two or more ESP32 microprocessors.
TFT_eSPI.h	Library for the ILI9341 display to work.
SoftwareSerial.h	Very important library that allows for multiple serial connections. This is important as Kaifas HAN interface uses 2400 Baud, while the other libraries use 115200 Baud.
PubSubClient.h	Library that allows the ESP32 to publish and subscribe to MQTT brokers
WiFiManager.h	Library used to make Access Point and connect to the internet, while also storing parameters such as MQTT API token, password etc.
SPIFFS.h	Saves variables to flash memory, making it so that the values remain saved even after reset.

Appendix M. Hardware schematics

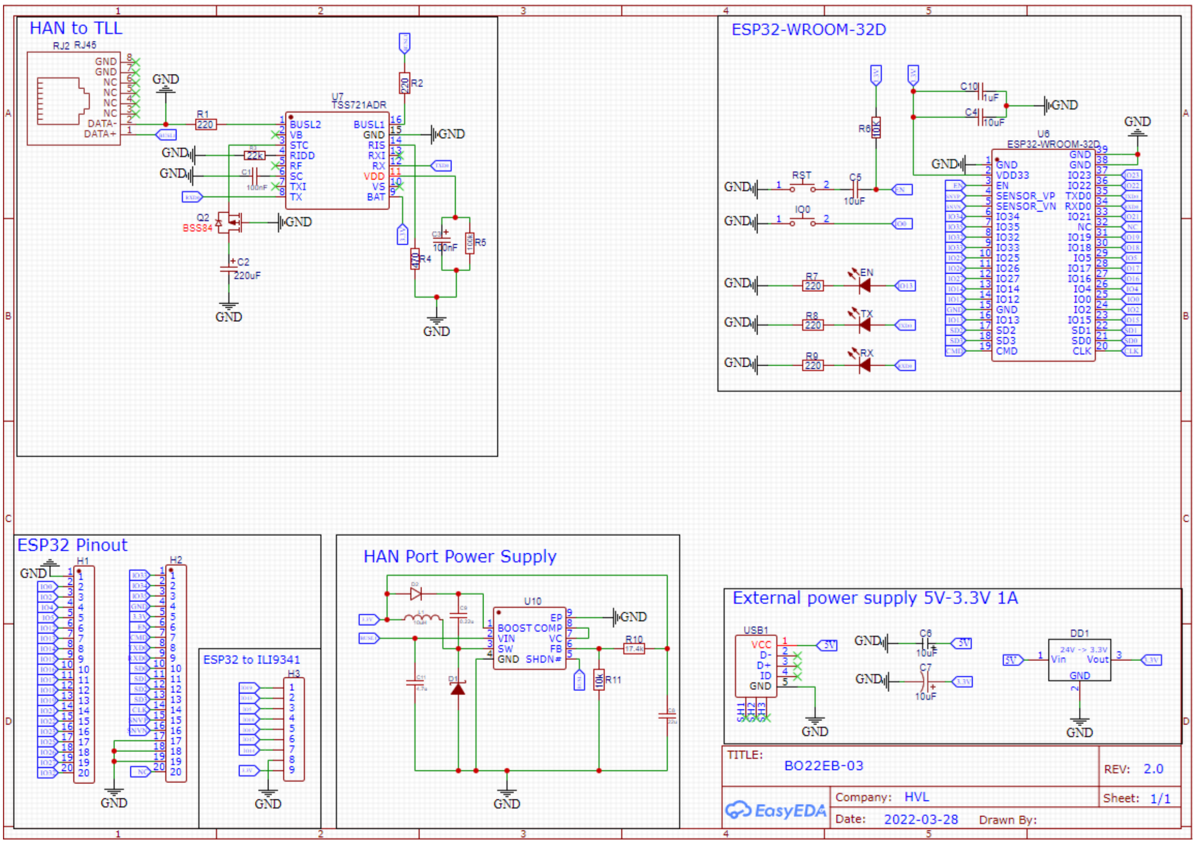


Figure 57 - Hardware schematic of complete circuit

Appendix N. Case Design

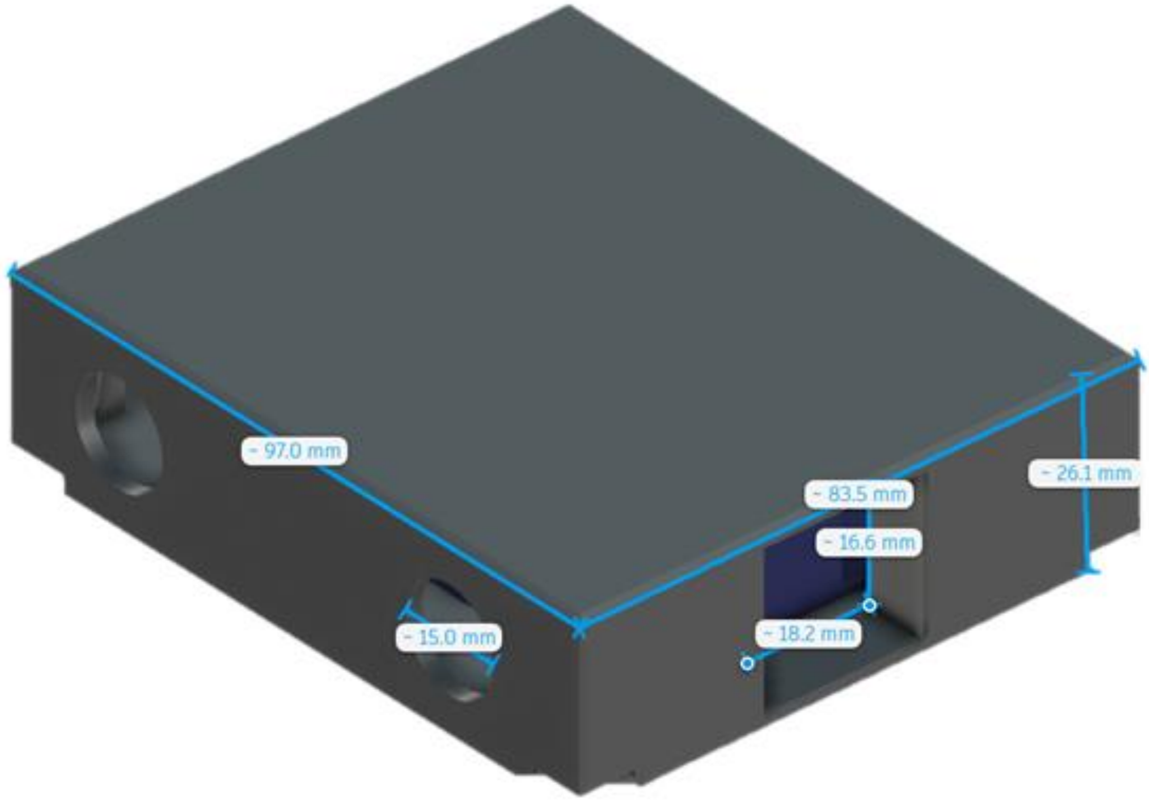


Figure 58 - SPR case design with clearer dimensions

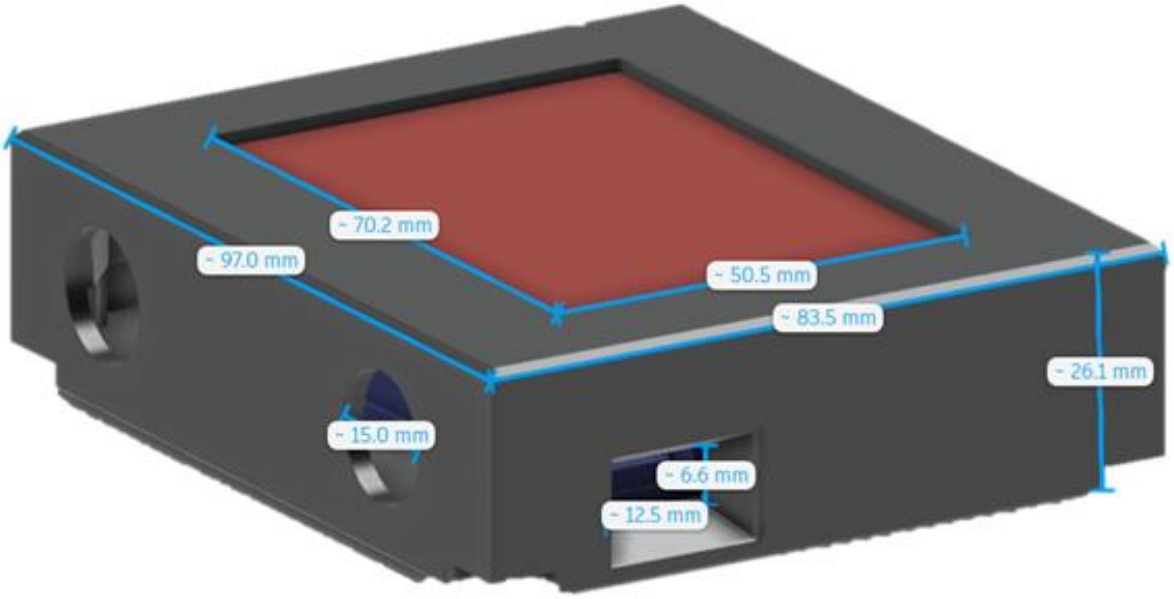


Figure 59 - SPD case design with clearer dimensions