UNIVERSITETET I AGDER

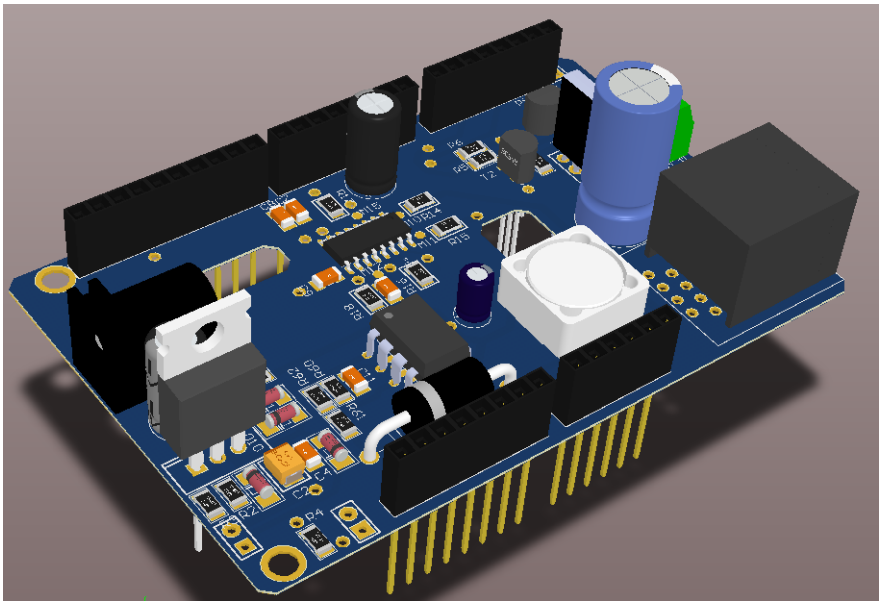
SmartHub with AMS interface

of

Sondre Håverstad, Jan Roar T. Mydland and Eivind Stendal

Supervisors: Geir Jevne and Ken Henry Andersen

Bachelor's thesis in ELE301, 6^{th} semester, e.g. Spring 2017



Faculty of Engineering and Science

University of Agder

Grimstad, May 2017

Status: Final

<AMS interface, SmartHub, nRF52, Meter-Bus, FreeRTOS>

UNIVERSITETET I AGDER

Summary

This report surrounds the introduction of the AMS smart meter, which is currently being installed in every Norwegian household. The AMS smart meter will collect measuring data about your power consumption about every few seconds and save the information by uploading it to a central location. The information will be used to monitor the power flow in terms of making the whole billing process automatic and improving the power grid. But it is hard to see what the user get out off it, except of a higher electricity bill.

This project tent to make the AMS a benefit for the user as well as for the power grid, by using the same information as the power companies to even out power peaks, which may result in higher power prices in the future. The group has put a lot of hours and effort into this project, and it has challenged us in finding the best solution to the problem. There has been done a lot of research to raise the knowledge about the AMS, but the AMS firmware could not be obtained. We have created a smarthub that uses the instantaneous power provided by a different power meter to calculate priorities with an algorithm, which processes this information with possibilities for extensions. The SmartHub can communicate with up to 7 heat devices and regulates them simultaneously by using the algorithm and to even out the consumption.

Power control by regulating devices is not a new technology, but the interface against the AMS is relatively new and there is little specific information about the measuring information to be found. The group came up with working result for a smart meter and this has taught us a lot about programming, research and PCB design.

Version Control

Version[1]	**Status**[2]	**Date**[3]	**Change**[4]	**Person**[5]
0.1	Draft	2017.01.09	Started on introduction	E.S, S.H, J.R.M
1.0	Final	2017-05-13	Final product	E.S, S.H, J.R.M

Table 1: Version Control

[1] **Version** Shows the Version number from 0.1 to 1.0 which is the last for the Final report.
[2] **Status** is Draft, Research or Final
[3] **Date** ISO format: yyyy-mm-dd
[4] **Changes** describes changes made since the previous version
[5] **Author** is the one who made the changes

Preface

This report is a product of a bachelor thesis in ELE301 at the University of Agder. The task "AMS interface for charging the blue battery" is one out of three bachelor thesis regarding "Charging of the blue battery" [7.2] given by assistant professor Geir Jevne. It is solved with NRF52 microcontrollers and new hardware that is required.

This report is intended for c programmers and electronic engineers working with smarthub solutions.
We would like to say thank you to assistant professor Geir Jevne and assistant professor Ken Henry Andersen for helping us as our supervisors with equipment and knowledge.

Grimstad 15. May 2017 Eivind Stendal, Jan Roar T. Mydland and Sondre Håverstad.

..

Contents

1 Introduction

1.1 Background

In the autumn 2016, power companies started installation of the new AMS smart meters after orders from Statnett and Norwegian Water Resources and Energy Directorate(NVE)[1]. AMS meters will replace all the conventional power meters that is connected to the Norwegian power grid, there are approximately 2.9 million power meters that are to be replaced, spread over 111 different power firms[1]. According to NVE 15-20% of the meters are to be replaced by the end of 2016, approximately 70% during 2017 and with the remaining 10-15% before January 1. 2019, with an estimated investment cost of 10 billion NOK[1].

The AMS meter will measure the instantaneous power consumption, production and power quality in each individual measuring point with a fixed interval[2]. Mainly done to get a better view of the power consumption in Norway, and making troubleshooting easier. This information shall also be available for 3rd party developers[2].

The power consumption in Norway is not even throughout the day, and the power distribution net must be designed for the peak periods[3]. By implementing AMS the power distribution companies can have different electricity price during a day, and they can make it more economical for the customer to have a lower consume in the peak periods, which can slow down the expansion of the power network. The power companies may also start with over consumption price as they have done before, a higher consume price if you are over a certain limit. This will hopefully make people more aware of how much power they are consuming.

1.2 Problem statement

The problem will consist of a system that controls devices based on the instantaneous power that the AMS smart meters provides, and make the house self adapt to the access of power. The system should lower the power peaks of the system and make it more constant throughout the day. The central device in this report will be called the master.

Main goals:

M_1: A central device should read instantaneous power consumption from AMS.

M_2: Use the instantaneous power consume to regulate minimum one heat source.

M_3: The central device should control the heat sources based on priorities.

M_4: Propose a communication solution between central device and heat sources.

Sub goals:

S_1: The central device should communicate with more then one heat source.

S_2: The central device should have information about the temperature related to the heat source.

S_3: The central device should have an user interface for configuration.

Expected result The group expects to finish the main goals and the sub goal S_1 and S_2. If the work is going faster then expected, S_3 will be started on as well.

1.3 Literature review

The most used literature in this report is from The Norwegian Water Resources and Energy Directorate[1][2]. Regarding the software research FreeRTOS [4] and Nordic Semiconductors [5] homepage are well used as references.

1.4 Prerequisites and Limitations

The project has encountered problems with the AMS smart meter. Agder Energy has informed us that we will not get access to the firmware before mid May, because it is still under testing and not officially released. This means that we must come up with an alternative solution otherwise the main goal of the project is lost.

The main goal M_4 is about sending a data packet between central device and heat devices and not how it is sent, which means the project may not include a optimal communication protocol. For instance, security in the protocol will not be considered.

1.5 Problem solution

The system shall move the consumption of the system from the peak-periods to the non-peak periods without reducing the comfort of the customer. A micro-controller will be used to regulate the system. Since we are not able to receive an AMS in time, the master has to obtain the data from a power meter with an Meter-Bus(M-Bus) interface.

The central and heat devices will be implemented with a micro-controller that supports a wireless technology. Under in figure 1 there is an example of how the system could be implemented in an apartment. In this figure 1 heat devices will be called slaves and the central unit are called smarthub. Typical devices that will be controlled by a slave are heat devices like electric radiator, underfloor heating and boiler. The heat devices and central/smarthub should be powered by the net and not by batteries.

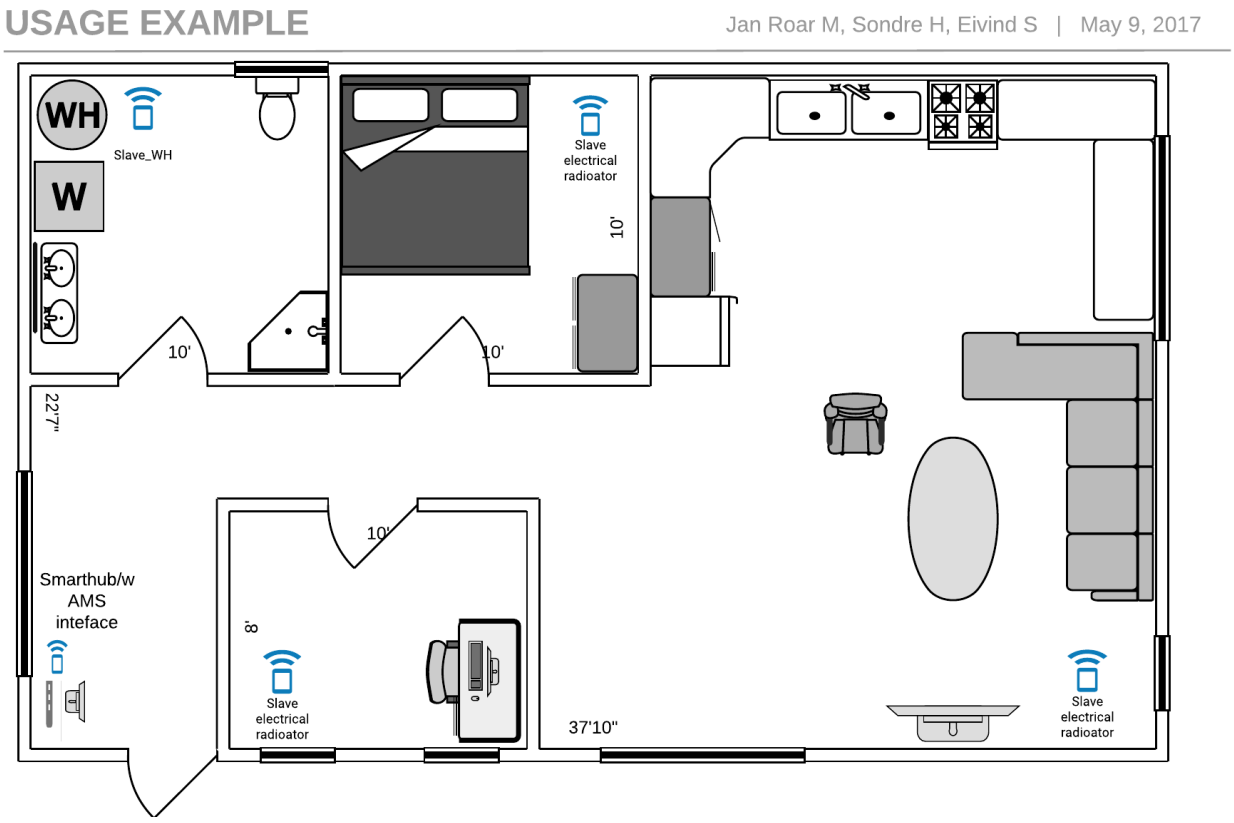


Figure 1: Usage example

1.6 Project plan

In the figure 2 we can see the work plan with calculated hours, there has also been made a time-line Gantt diagram, in the appendix. see figure 59.

Task	Date		Persons	Days	SUM (Hours)
	From	To			
Feasibility study	16.jan	24.feb	2	30	450
Mbus tranceiver					
Hardware (Design, milling, testing and completion)					
Design a mbus tranceiver.	27.feb	13.mar	1	10	75
Software (Develope, debug and testing)					
Develope code, UART, MBUS tarnceiver	06.mar	13.mar	2	6	90
Integrate OBIS codes for reading AMS	27.feb	06.mar	2	6	90
Report writing	13.mar	15.mar	3	3	67,5
Slave controller					
Hardware (Design, milling, testing and completion)					
Create a switch and read temperature	20.mar	23.mar	1	4	30
Software (Develope, debug and testing)					
Develope code for sending/receive info	20.mar	23.mar	2	4	60
Report writing	24.mar	24.mar	3	1	22,5
Master controller					
Hardware (Design, milling, testing and completion)					
Create a solution for measuring ampere	14.mar	23.mar	1	8	60
Build a 3.3V supply	23.mar	31.mar	1	7	52,5
Software (Develope, debug and testing)					
Develope/Integrate protocol into RTOS	31.mar	17.apr	1	12	90
Integrate Mbus protocol Uart Master	17.apr	04.mai	2	14	210
Developing of freeRTOS	31.mar	11.apr	2	8	120
Ammeter reading	24.apr	04.mai	1	9	67,5
Report writing	05.mai	10.mai	3	4	90
Compleation of report	10.mai	15.mai	3	4	90

Calculated hours of work each day		7,5
Calculated work hour inkluded feasibility study	130	1665
Calculated work hour whithout the feasibility study report = 13 weeks	100	1215
Calculated working hours for each group member, per week		31

Figure 2: Work plan

1.7 Report outline

The rest of the project structure is as follows:

2 Theoretical background
Chapter 2 contains research and background knowledge that concerns the solution in chapter 3.

3 Solution
This section is divided into 4 subsections requirements, design specification, implementation and validating & testing. The Requirements describes functional and non functional requirements that will most likely be implemented in the project; Design specification describes how the problem will be solved with simple diagrams and text, it answers to the research in chapter 2 Theoretical background; while the implementation describes how the problem is solved.

4 Discussion
The discussion will be about the project solution, main goals, sub goals, what we have learned and what we could have done differently.

5 Conclusion
This section includes a summery of the task and how the group archived based on the main and sub goals.

2 Theoretical background

2.1 Smart power meter(AMS)

The new AMS smart power meters shall replace the existing analogue meters in today's power grid[1]. AMS measures instantaneous power at equal intervals and saves the values at approximately every 10 seconds [2]. This information is logged and uploaded to power companies own server at intervals of 10 to 60 minutes.

In addition to reading the instantaneous power, the meter will be to record voltage and current at the inlet of the house separately, and along with the measured values of all three phases with the same interval. This will help make troubleshooting power lines easier and more efficient; ensure voltage quality to the end user; Makes it possible to detect ground faults in all subsystems connected the AMS meter, in order to save power consumption and increase efficiency in the power grid. [2]

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 through the internet (mobile app) and a standardized electrical interface (EN 13757-2) [2]. Through this connection also called HAN interface (ISO/IEC 8877) the customer shall be given access to the information mentioned earlier, free of charge [2]. This information will be used in this project to develop a system that reduces power peaks and to equalize the power consumption throughout the day.

Figure 3: Kamstrup AMS smart meter [6]

2.2 M-BUS Tranceiver

To receive information, AMS meter offers the HAN interface ($ISO/IEC8877$). In order to read this information there must be made a receiver that can interpret the M-Bus protocol ($EN137572$), which is sent out on the interface [2].

M-BUS is a German developed serial bus protocol, specially designed considering the remote reading of measurements, in this case the power consumption. It is designed to keep a periodic reading of the devices connected, such as power meters in several houses [2].

The information sent from 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 [7]]. These OBIS codes will be equal on all AMS meter installed, to ensure that customers can use the same reading device regardless of AMS meters used [2].

Based on this information and since it has not yet been created a product that does the job. There must created a M-Bus device that can receive this information and by using OBIS codes, convert this information into something the microprocessor can understand.The developers of M-Bus has developed in cooperation with Texas Instruments a component (TSS721A)[8] containing core logic to a M-Bus transceiver that is a good starting point and something to take advantage of in this task[9].

2.3 Singlephase energy meter with M-Bus interface

Figure 4 shows the energy meter that is used in this project. Despite the small drawback that this is a lighter version of an energy meter then the AMS smart meter that was intended to be used. It still offers M-Bus protocol interference. This energy meter gives information about power (Total and Partial), voltage, current, active and reactive power [11]. In this project the information about active power is the most interesting. The energy meter also has a feature that saves all the data/registers in case of a power failure.

Figure 4: Energy meter with M-BUS interface [10]

2.3.1 Connecting via M-BUS

Following information about the energy meter are collected from the data sheet of the meter [11]. The energy meter offers automatic detection of transmission rates, although the transmission rates have to be either 300, 2400 or 9600 Baudrate. The energy meter does not respond to unknown queries. The following telegrams are supported:

- *Initialisation*—SND_NKE—Response:0xE5
- *Reading meter*—REQ_UD2—Response:RSP_UD
- *Changing primary address*—SND_UD—Response:0xE5
- *Reset* T_{part}—SND_UD—Response:0xE5

In this project there is only *Initialisation* and *Reading meter* that is used, these two are the only telegrams that are described in more detail underneath here.

Initialisation: The telegram structure SND_NKE is used in order to initialise the energy meter. Figure 5 shows a detailed description of the actual telegram structure. The response from the energy meter is expected to be hexadecimal value of $E5$. More detail about SND_NKE is found in next section 2.4.

Telegram structure (detailed)

Byte	Value	Description
1	0x10	Start
2	0x40	Send or reply, reset
3		Primary address
4		Checksum
5	0x16	Stop

Figure 5: SND_NKE telegram structure in detail [10]

Reading response: The telegram structure REQ_UD2 is used in order to read out the values from the energy meter. The response from the energy meter is sent in a RSP_UD telegram structure, as seen in figure 6. Also notice that all the values are sent in one packet after sending an request. More detailed information of the telegram structures REQ_UD2 and RSP_UD can be found in the next section 2.4.

Telegram structure

0x68	0x38	0x38	0x68	0x08	PAdr	0x72	ID	0x43	0x4c	DEV
02	ACC	STAT	0	0	0x8c	0x10	0x04	Eto	0x8c	0x11
0x04	Epa	0x02	0xFD	0xC9	0xFF	0x01	V	0x02	0xFD	0xDB
0xFF	0x01	I	0x02	0xAC	0xFF	0x01	P	0x82	0x40	0xAC
0xFF	0x01	Pr	CSum	0x16						
	Variable at 1, 2 or 4 bytes									

Figure 6: Request from energy meter (RSP_UD) [10]

Figure 7 shows a more detailed figure of the telegram structure in figure 6.

Telegram structure (detailed)

Byte	Value	Description
1	0x68	Start
2	0x38	L_Read
3	0x38	L_Read
4	0x68	Start
5	0x08	C
6	x	Primary address
7	0x72	CI
8	x	ID1 (LSB)
9	x	ID2
10	x	ID3
11	x	ID4 (MSB)
12	0x43	MAN1
13	0x4C	MAN2
14	x	DEV (Typ - Version)
15	02	MED (Electric)
16	x	ACC
17	0x01 0x02 0x04 0x08 0x10	STAT Application_busy Any_Application_Error Power_low Permanent_Error Temporary_Error
18	0	SIG1
19	0	SIG2
20	0x8C	DIF
21	0x10	DIFE
22	0x04	VIF (0.01 kWh)
23	Eto_4	T1 total
24	Eto_3	
25	Eto_2	
26	Eto_1	
27	0x8C	DIF
28	0x11	DIFE
29	0x04	VIF (0.01 kWh)

Byte	Value	Description
30	Epa_4	T1 Partial
31	Epa_3	
32	Epa_2	
33	Epa_1	
34	0x02	DIF
35	0xFD	VIF
36	0xC9	VIFE (1V)
37	0xFF	VIFE
38	0x01	VIFE
39	V_2	Voltage
40	V_1	
41	0x02	DIF
42	0xFD	VIF
43	0xDB	VIFE (0.1 A)
44	0xFF	VIFE
45	0x01	VIFE
46	I_2	Current
47	I_1	
48	0x02	DIF
49	0xAC	VIF (0.01kW)
50	0xFF	VIFE
51	0x01	VIFE
52	P_2	Power
53	P_1	
54	0x82	DIF
55	0x40	DIFE
56	0xAC	VIF (0.01kVAR)
57	0xFF	VIFE
58	0x01	VIFE
59	Pr_2	Reactive power
60	Pr_1	
61	CS	Checksum
62	0x16	Stop

Figure 7: More detailed figure of the request from energy meter (RSP_UD) [10]

The next figure 8 shows where the different values are stored, how they are stored, and also what unit the values are in.

Byte	Content	Type	Description
23 - 26	Eto=x	4 b. BCD	Energy total
30 - 33	Epa=x	4 b. BCD	Energy partial
39 - 40	V=x	2b. Integer	Voltage
46 - 47	I=x	2b. Integer	Current
52 - 53	P=x	2b. Integer	Power
59 - 60	Pr=x	2b. Integer	Reactive Power

Unit with multiplier		ALD1
I	(Current)	0.1 [A]
U	(Voltage)	1 [V]
P_{active}	(Power)	0.01 [kW]
$P_{reactive}$	(Reactive Power)	0.01 [kVAR]
E	(Consumption)	0.01 [kWh]

Figure 8: Information about where the values are stored) [10]

The values of power which is the most interesting in this project are found in byte 52 and 53, also worth to mention that the resolution is 10 W pr bits.

Changing M-BUS address: The default setup of the energy meter is typical with the primary address set to 0 ($0x00$). One possible way to change the address is to change it directly on the device itself. The following description is found in the energy meter documentation [11].

- In the menu, go for "U"
- Push long ($\geq$ 3 sec) $\rightarrow$ "$MBUS - ADR$"
- Push short $\rightarrow$ M-Bus address +1, push long $\rightarrow$ M-BUS address +10
- Once the desired address is selected, wait until the root menu comes back to validate

The other possible way is to use the *Changing primary address* telegram structure. This is done by sending a SND_NKE telegram structure to the energy meter. A more detailed description of the SND_NKE structure can be found in the next section 2.4.

2.4 M-BUS protocol

Meter bus (M-BUS) was specifically designed for remote reading of measuring equipment, and supports up against hundreds of devices connected to the same bus. M-Bus consists of one master and several slaves connected in parallel with each other over a two-wire bus and reads measurements periodically [9]. The slaves can't communicate with each other, so there has to be a Master-Slave structure. M-BUS support remote powering of slaves and works in a (Half Duplex) fashion. The wire named (MBus+) is kept at constant 36V in order to supply the slaves with power. The other wire (Mbus-) switch between 0 and 12V, giving either 36V or 24V. The difference between (MBus+) and (MBus-) determines if it is a one or zero being transmitted. 36V between them corresponds to a logical "one" and 24V to a logical "zero". This gives a improvement range, because the devices do not care if the bus voltage at logical "one" is 36 or 19. The receivers only sense and reacts to a change in 12V between the wires.

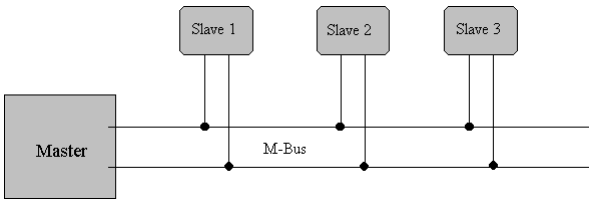


Figure 9: Block diagram M-BUS [9]

Protocol: Underneath here comes a description of the protocol, this information is based on information found in the M-Bus description documentation [9]. The protocol that is used between master and slaves is a asynchronous serial bit transmission, which is equivalent to the UART protocol described in the next section [2.5]. Figure 10 shows an example of a transmission of one character from the master to a slave.

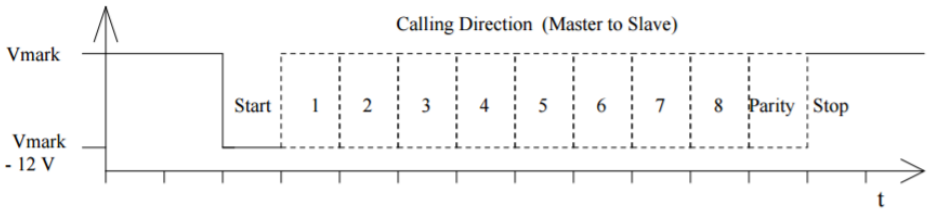


Figure 10: Transmission of a character in calling direction [9]

Some requirement must be taken in order to get the transmission to work. There must not be a pause in between two telegrams. The second requirement is that the eleventh bit, which is the "stop" bit, must be a logical 'one', equivalent the "start" bit has to be a logical 'zero', in order to start transmission. As seen on the figure 10 the bits are sent in an ascending order, least significant bit first (LSB).

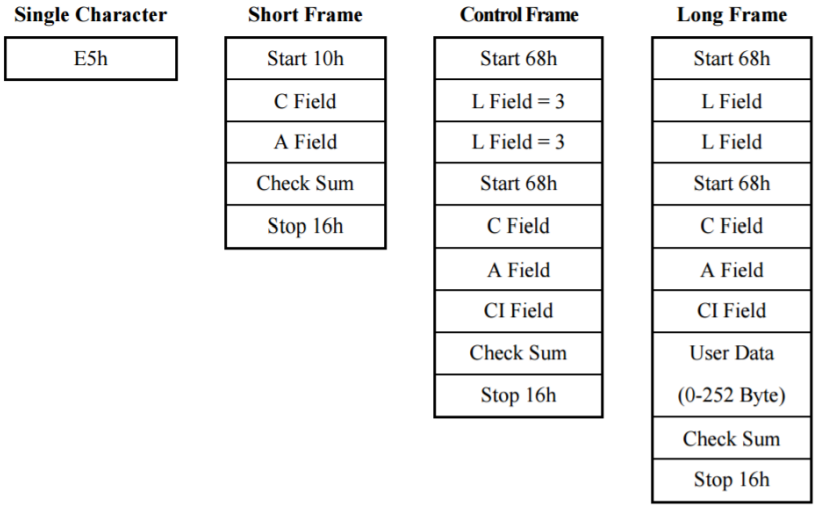


Figure 11: Telegram frames [9]

There are four different telegram frames that is used in the communication between a master and the slaves in a M-Bus system.

- **Single Character:** Consist of just one character, typically hexadecimal value $E5$. Typical usage of this is to acknowledge a receipt of transmission.
- **Short Frame:** Frame with a fixed length, starts with the hexadecimal value of 10 and stops with the hexadecimal value of 16. In between these two we find the C-field, A-field and a check sum. The check sum can be found by adding C-field and A-field with each other. This telegram structure is used when sending a SND_NKE telegram.
- **Control Frame:** This frame is just a short version of the Long Frame. With a fixed L_field of 3. This telegram structure can be used when sending a SND_UD or an RSP_UD.
- **Long Frame:** The long frame starts with the hexadecimal number 68 followed with the length (L field) that is sent twice, then the start value of $0x68$ is sent one more time. L field is build up with the user data plus 3. Then comes C field, A field and CI field. The checksum is build up over the same area as the length field, and the last byte to be sent is stop with a hexadecimal value of 16. This telegram structure is used in the same way as the control frame, that means that it is used when sending SND_UD or RSP_UD telegrams.

The different fields: Here comes a short description of the different fields in the telegram structures. All fields have a length of 1 byte, corresponding to 8 bits. This description is based on information found in the M-Bus description documentation [9].

- **C field** The C field is also called Control field or function field, this field specifies the direction of the data flow. It is also responsibly for various tasks in both the calling and replying direction.

Bit Number	7	6	5	4	3	2	1	0
Calling Direction	0	1	FCB	FCV	F3	F2	F1	F0
Reply Direction	0	0	ACD	DFC	F3	F2	F1	F0

Figure 12: C-Field [9]

Figure 12 shows a more detailed overview over the C-field. Bit 7 is not used yet, and bit 6 is used to indicate the direction of data flow. FCB bit can be used to indicate successful transmission, if the expected reply is faulty the master sends a new telegram with FCB bit set to 1. FCV is set to 1 if FCB is used. In replying direction, the slave uses ACD to indicate transmission of Class 1 data. DFC (data flow control) is used to tell master that the slave can't accept no further data.

- **A field** This is the address field, which is used to identify the slaves. Addresses from 1 to 250 can be used, address 254 and 255 is used in broadcasting to the slaves.
- **CI field** The CI field is called the control information field, used in the telegram structure in order to distinguish between the formats of the long and control frame. Also implements variety of different actions in the master and or the slaves.
- **Check sum** This can be used to recognize transmission and synchronization faults. Calculated by adding the different fields mentioned above here, not taking care of carry digits.

Underneath here is a figure 13 of the control codes used in the M-Bus protocol. The interesting part as seen in the previous section 2.3 is SND_UD, REQ_UD2 and RSP_UD.

Name	C Field Binary	C Field Hex.	Telegram	Description
SND_NKE	0100 0000	40	Short Frame	Initialization of Slave
SND_UD	01F1 0011	53/73	Long/Control Frame	Send User Data to Slave
REQ_UD2	01F1 1011	5B/7B	Short Frame	Request for Class 2 Data
REQ_UD1	01F1 1010	5A/7A	Short Frame	Request for Class1 Data (see 8.1: Alarm Protocol)
RSP_UD	00AD 1000	08/18/28/38	Long/Control Frame	Data Transfer from Slave to Master after Request

Figure 13: Control codes of the M-Bus protocol [9]

2.5 Universal Asynchronous Receiver/Transmitter(UART) protocol

As seen in previous section 2.4 the M-Bus protocol is equivalent to the UART protocol. Underneath here follows a small description of the UART communication protocol, this information is based on information found in *All about Circuits* site regarding the UART protocol [12]. UART is a robust, moderated-speed and full-duplex communication protocol. Basic UART only consist of three wires. TX transmission serial data, RX received serial data and ground. There is no need for a clock signal, the receiver and transmitter is set up to communicate on a specific clock signal. Figure 15 shows a typical UART sending of a byte. Consisting of a start bit, stop bit and 8 bits data which is equal to a byte.

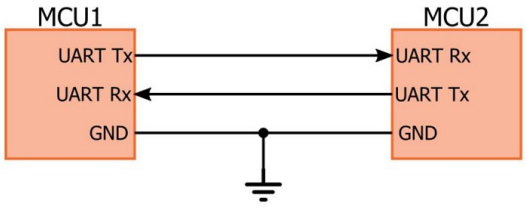


Figure 14: Block diagram of UART [12]

Figure 15: Sending of a byte [12]

- **Start bit** This bit indicates the start of a one-byte sending. Typical this bit is logic low, because the idle state is typical logical high.
- **Stop bit** This bit is typical logical high, the same as the idle state.
- **Baud rate** This is the approximated rate that the data can be transferred at, referred too as bits per second.
- **Parity bit** This bit is used in error detection. There are two types, odd parity and even parity. Odd parity is logical high if the data byte contains an even number of logical high bits, and even parity is the opposite of that.

2.6 Protocol research

This section is about the communication protocol between the heat device and the master/ central device.

A system that is easy to implement in an existing building without new wiring is necessary to keep the cost and work to a limit. This is a small recap of the most popular protocols in 2017.

2.6.1 Zigbee

Zigbee is a protocol using radio IEEE 802.15.4 wireless technology to communicate. It is a widely used protocol among many producers in the world. What makes Zigbee a relevant protocol is the ability to connect in a mesh network with low power consumption with unlimited devices [13]. In a report comparing other protocols the zigbee uses around 185 μJ/bit, 0.035706W when transmitting 24 bytes of data and sends 192 bits/s[14]. ZigBee offers the opportunity to connect energy harvesting product and self powered devices.

- Energy = 185 μJ/bit
- Power consumption = 0.035706W when transmitting 24 byte
- Bits per second = 192 bps
- Distance 10-100 meters

2.6.2 Z-wave

Z-wave is based on the same concept as Zigbee, but the background of the idea is to make it simpler and cheaper, based on plug and play system [15]. Z-wave is today the leading technology in wireless home control with over 1700 products [16]. The mesh of Z-wave device can have up to 232 devices and they can all be used as repeaters except for those battery devices. This is because a device cant sleep if used as a repeater and therefore uses more power. Z-wave also support IP architecture.

2.6.3 WIFI

WIFI is already integrated in most houses which makes a good base. Laptops, mobile devices and game consoles are using this technology and it is working great for there purpose. However WIFI is not a low energy protocol that works perfectly for low energy devices. It uses 0.00525 μJbit which is very efficient, however current consumption does not reduce when throughput is reduced. WIFI uses around 0.210 W for sending one UDP packet of 40Mbps [14]. This is much higher then other compared protocols [14].

- Energy = 0.00525 μJbit
- Power consumption = 0.210 W when transmitting 40MB
- Bits per second = 40 Mbps
- 150 M

2.6.4 Bluetooth 4 low energy

Bluetooth technology is a known protocol among most people, used for personal connectivity. The newer version bluetooth low energy(BLE) is a low energy protocol and it is very energy efficient. It was not made for smarthome applications and for that reason it is maybe not suited to the job as good as other protocols. However Bluetooth are working on a prototype with mesh network specifications, and this combined with the efficiency of BLE can be a good combination [17][14].

- Energy = 0.153 μJ/bit
- Power consumption = 0.147 mW
- Bytes per second = 960 bps
- Range 10-30M

2.6.5 Thread

Thread is a brand new protocol from 2014 founded by 7 companies including Google and Samsung. Based on IEEE 802.15.4 RF which is the same as Zigbee but they focus on the 6LoWPAN which Zigbee only got as an upgrade. There main task was to make an efficient, simple and trustworthy system that could handle over 250 devices [18].

A new smart mesh control makes it possible for nodes to go in sleep mode and not having to check in. Messages for sleepy nodes is buffered by their parents and sent when they wake up [18].

2.6.6 Universal powerline bus

Universal powerline bus(UPB) is a protocol that communicate through the powerlines and can have up to 250 units and a range over one mile [19]. Every UPB central can have 250 different network ID to differentiate from the neighbour. If you and your neighbour got the same network ID you can control each others devices which is not to prefer [15]. A simple ID configure is all it takes and the system should not interact with any other systems unless there is 250 other close systems [18].

2.6.7 Insteon

Insteon is a protocol that uses powerlines and wireless connection. The protocol can have unlimited devices connected and are only limited by memory. It uses mesh network with both wire and wireless connections. By using bridges you can connect with many other types of systems [20].

2.7 The nRF52

Nrf52 is a powerful ultra-low-power "system-on-chip" from Nordic Semiconductors and built for the IoT market . The nRF52 is a microcontroller build around a 32-bit ARM Cortex-M4 CPU with 512kB + 64kB RAM and with a built-in BLE Transceiver with a lot of functions making it perfect for many applications. The microcontroller also comes with a SDK with many good functions and examples[21].

2.8 Design patterns

There are several ways to implement the software design patterns. A small description of some different patterns can be found underneath here.

2.8.1 Event driven

In an event driven programming the flow of the program is determined by the occurrence of events [22]. The software or microprocessor typically waits in a low power state until an event occurs (could typical be a hardware interrupt). Then a callback or event handler function will be called to process this event, after the event it returns to the waiting state. Figure 16 shows a small example of an event driven software.

Most of the SDK examples from Nordic Semiconductors are based upon the event driven algorithm [23].

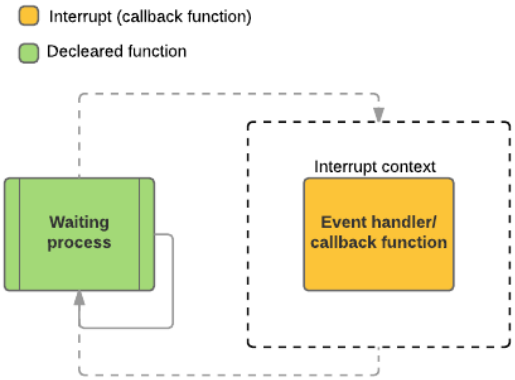


Figure 16: Example of an event driven software

2.8.2 Finite state machine

This can also be referred to as FSM. The webpage *All about circuits*[24] have a net way to explain what FSM is; "*A finite state system is a system where only a set number of real, defined states can exist*". Another way to explain this could be to say that no matter where the code is, it will always end up back to a known state again. In this patterns it's easy for the programmer to force the software to execute in a specific order.

Figure 17 shows a very simple example on how this software pattern could be implemented. This could be a simple program to turn on and off Light Emitting Diodes, and the received signals could be a switch that was pressed.

FSM could of course be implanted inside a function as well.

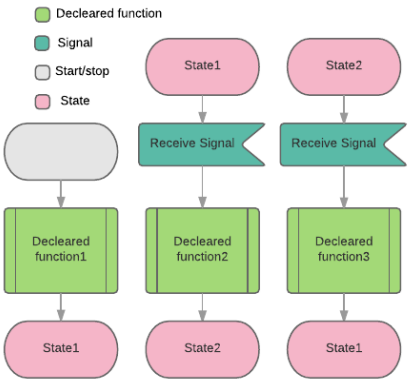


Figure 17: Example of an Finite State Machine software

2.8.3 Multitasking

The concept of multitasking is to rapidly switch between tasks, to make it appear as if each task are executing at the same time [25]. A task is sometimes referred to as a thread.

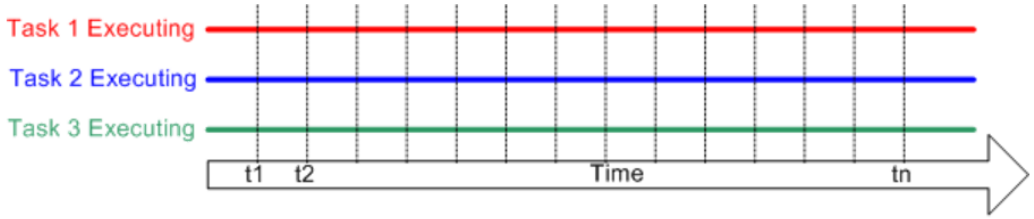


Figure 18: Multitasking [25]

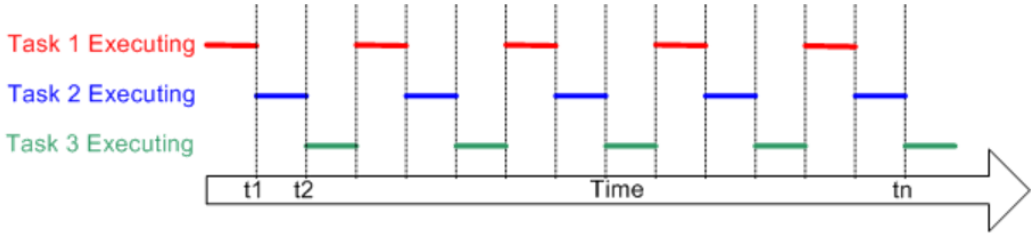


Figure 19: How multitasking works [25]

It is the job of the scheduler in an operating system to decide which task that is allowed to run at a specific time, and to switch between the different task [26].

RTOS: A Real Time Operating System (RTOS) is designed to provide a predictable execution pattern. This is especially of interest in an embedded system, where there often is a real time requirements. This means that the scheduler in a RTOS is responsible to execute a task in a specific time after an event has occurred [26].

2.8.4 FreeRTOS

One type of RTOS that is small enough to run on a microcontroller is FreeRTOS. FreeRTOS provides a real time scheduling functionality, inter-task communication, timing and much more. See FreeRTOS web page for more details [26].

Figure 20 shows the different states a task can have, this information and a more detailed information is found at [27].

- **Running:** The task that is currently executed is located in this state, and there can only exist one task in this state (if the processor just consist of a single core).
- **Ready:** Task that has been unblocked is placed in this state, and they are waiting on a higher priority or equal priority task that is already in the running state.
- **Blocked:** A task that is waiting on a event to occur is placed in this state. Typical a task is blocked to wait for a queue, semaphore, event group, notification or semaphore event.

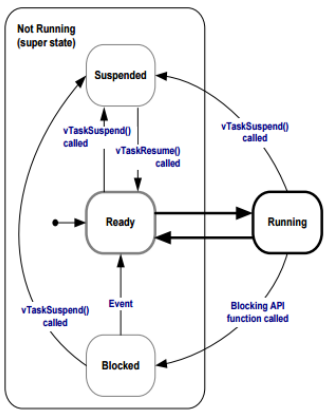


Figure 20: Task states in FreeRTOS [27]

- **Suspended:** In these state only tasks that has called vTaskSuspend() is placed, and in order to exit this state vTaskResume() is called.

More details about FreeRTOS Underneath there is a list off more relevant information regarding FreeRTOS such as Heap memory, tasks, queues, binary semaphores, mutexes and event groups. All of this information and much more detailed information is found in [28].

- **Allocating heap memory:** Figure 21 shows in A the total allocated memory that is defined in configTOTAL_HEAP_SIZE. In B one task is created and this takes up a small part of the total heap memory. And in C there are three task that has been created.

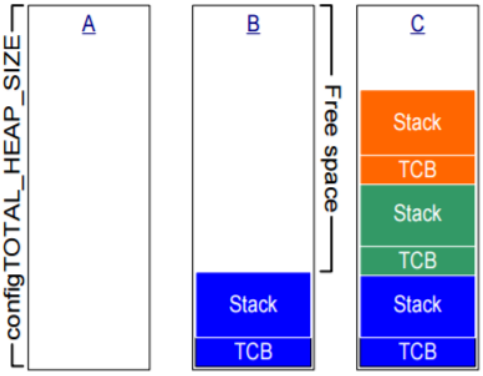


Figure 21: Visuel heap memory allocation [28]

- **Tasks:** Each task is a small program, implemented as a C function. They must only return a void and take a void pointer parameter. A task will normally run forever in an infinite loop, and will not exit. A single definition of a task can be used to create any number of tasks.
- **Queue:** A queue is used to hold a finite number of fixed size data items. These could either be a copy of the data, or refereed as a pointer to a specific data.
- **Binary semaphore:** This is used to implement synchronisation between tasks, or between a task and an interrupt. This can be seen as a queue with only one item.
- **Mutex:** Mutexes is used to guard a precious resource.
- **Software Timers:** A software timer is used to execute a specific function at a set of time. More specific a callback function will be executed when the timers period expires. A software timer could be either be a one-shot timer or a auto-reloaded timer.

2.9 Powering and loads

2.9.1 Switch mode power supply

Switch mode power supplies are widely used and often replacing linear AC to DC converters, a buck boost converter is popular because of it's efficiency. It uses a inductor to store energy and boost the voltage by switching current through it at high speed, the voltage builds up until the wanted output is reached, which is higher than the input voltage.

2.9.2 Loads

In a normal Norwegian household, the largest fuses for a normal wall socket can deliver 16 ampere(A), which gives 3600 watt(W) with 230 volts(V), from equation 1. In order to control a 3600W, we need a device that can handle such. Devices that can be used for this purpose and is able to break these loads can be created out of TRIACS in form of solid state relay, transistors, MOSFETS or relays. The three first options needs external circuits and heat sink in order to create a stable and viable circuit.

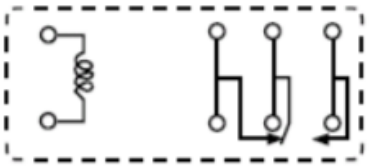


Figure 22: Relay [29]

$$P = U \cdot I = 230V \cdot 16A = 3600W \tag{1}$$

3 Solution

3.1 Requirements

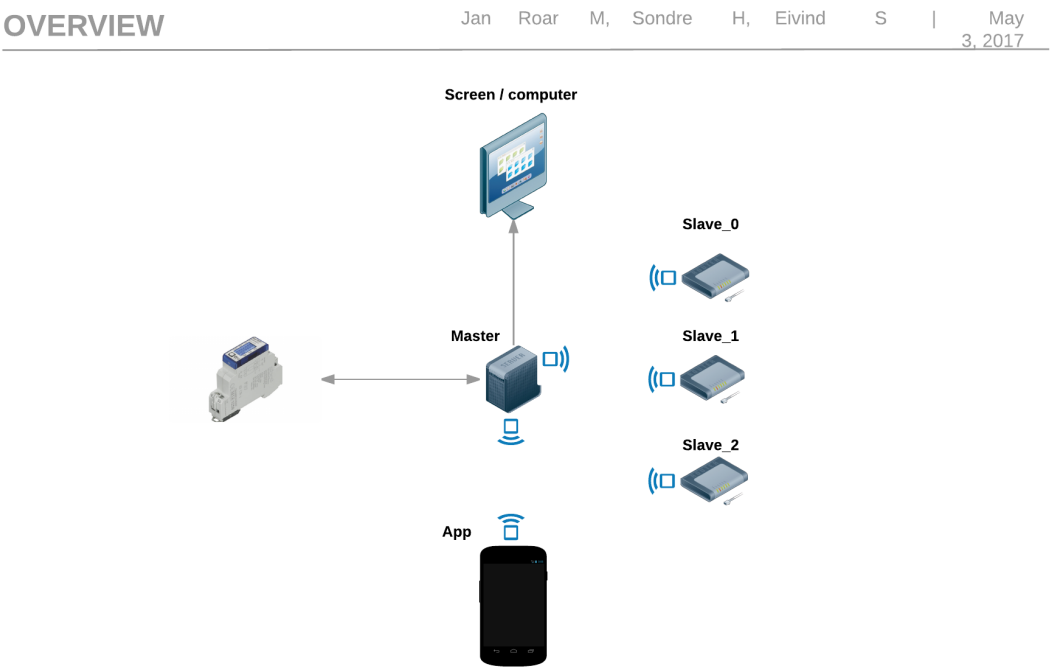


Figure 23: System overview

3.1.1 Functional requirements

- The master should read instantaneous power consumption from power meter with M-BUS(Changed from AMS).
- Use the instantaneous power consume to regulate minimum one slave device.
- The slaves should have priorities which the master can use for decisions.
- The master have an algorithm that should turn on and off slaves based on priorities, price(time) and power consumption.
- Slaves must have some sort of switch/relay in order to turn on and off devices.

3.1.2 Non functional requirements

- Master and slave should be able to get powered from 230VAC.
- Slaves must have the ability to read temperature and provide this to the master for prioritizing.
- The master should have some sort of M-BUS transceiver.
- The master should have the ability to receive configuration data from a user interface.

3.2 Design Specifications

3.2.1 Master - slave protocol

Choosing protocol There is many protocols to choose from with different specifications that are doing more or less the same job. The project need a standard that works great on small micro-controller beacons. After reading about the different protocols the most attractive protocols for the purpose is Zigbee, Z-wave and Thread. Z-wave is based on a plug and play system and may not be suitable for the purpose. Zigbee and Thread are attractive protocols because of the energy efficiency and mesh network. However when Bluetooth is launching their mesh network support it might compete with Zigbee and Thread.

For the task to be done there is a limited time of 3 months left which needs to be considered. The group has background-knowledge with Nordic Semiconductors nRF52 from previous project, and the software has not been released for Zigbee and Thread yet. Consequently the available nRF52832 with Bluetooth is chosen for the project [30].

SDK version The master can be connected with up to 8 BLE devices who communicates with the Nordic UART service. The newest 132 softdevice v4.0 from Nordic Semiconductors supports up to 20 peripherals and may be easy to implement, but it is not supporting freeRtos yet and will therefore not be implemented in our system. To choose a brand new SDK version also means less examples and tips. This taken into consideration the SDK 12.2.0 will be used in this project.

BLE Services The SDK V 12.2.0 comes with 34 premade BLE services [31]. The optimal solution would be to make our own service, however the BLE protocol is not the main task and therefore we choose to use a premade service. The group has some experience with NUS (Nordic UART Service) from last semester. NUS service is a solution that uses the UART principle over BLE and can send 20 bytes of data at once, which is more than enough for the given task. Security of the BLE will not be taken into account.

3.2.2 Hardware

This project will contain several layouts of Printed Circuit Board(PCB) to be constructed, one PCB for the master and at least one Slave PCB. The master or slave is complex, so every part of the design will be individually tested before they are merged together. The benefit of this is that it makes it possible to reuse the same circuits in both Master and Slave design and only create connection between the schematics. This will save both time and effort to complete the task in time.

Figure 24: PCB tools

Simulations will be done in **LTSpice** and the PCB layout will be done in **Altium designer** that the University of Agder have licenced, which both are familiar to us from previous work. As an extension to the PCB design, **Saturn PCB design** will be used as a tool for calculating trace thickness, spacing, via's and creepage distances and so on, to get the best design possible.

Master and the Slave PCB, will be based on a template for an Arduino shield provided by forum.arduino.cc, template to be found at: [32].

3.2.3 Master hardware specification

Master Hardware The Master main task is to collect the instantaneous power from the AMS smart meter(watt-meter w/M-BUS), calculate priorities and act by turning on/off or regulate Slave devices. To receive information from the AMS, the master need a M-BUS transceiver which converts UART messages to Meter-Bus(M-BUS) messages and present them to the watt-meter, which it will respond to. It would be beneficial if the Master could be supplied from the 230VAC.

In order to do that the design will be equipped with a 12V to 3V3 regulator which retrieves 12VDC from a 230VAC to 12VDC transformer as shown in figure 25, 3V3 will be provided to the nRF52 in order to power the master and make it independent of external power supply. The 12VDC will also be boosted up to a 34V by a buck boost converter to give enough voltage to the M-BUS, which will allow the communication between the watt-meter and the M-BUS transceiver.

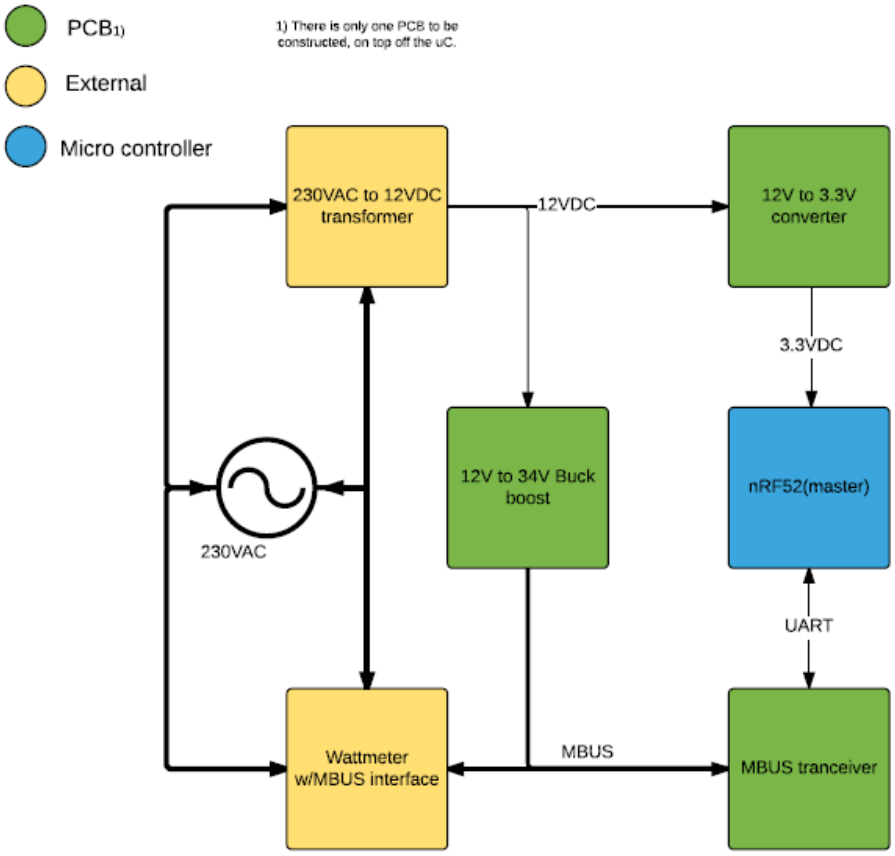


Figure 25: Master block diagram

3.2.4 Software tools

Compiler Keil MDK-ARM and IAR Embedded Workbench is the recommended integrated development environment(IDR) for nRF52 [5]. UiA got licenses on Keil and is therefore the used IDR for the project, but the licence is limited to 32kB hex files. The code might be bigger than 32kB and the solution to this is to program in Keil and compile with GCC. nRFgo studio will then be used to program the micro controller.

Analyze Digital signal analysis will be done by using Saleae[33]. UART will be used for the communication between power meter and master, and the nRF52 only got one UART instance. The logger function is also using UART which will make a conflict. Segger RTT will be used instead of the UART logging, this solves the problem[34].

3.2.5 Master software

Figure 26 shows a mind map of the functionality of the master design. The master will be implemented with a combination of threads using freeRTOS, events and state machine. Semaphores and mutexes will be used to avoid read/write conflicts in the program and queues will be used to send data from one thread to another. More information about different design patterns are found in 2.8.

To obtain the data from the M-BUS power meter mentioned in section 2.3 the master will use the UART protocol on the micro controller. There has been made a c-code file and header file regarding the different telegrams used to get connected to the M-BUS power meter described in section 2.3. The header file can be seen in appendix 7.6.6.

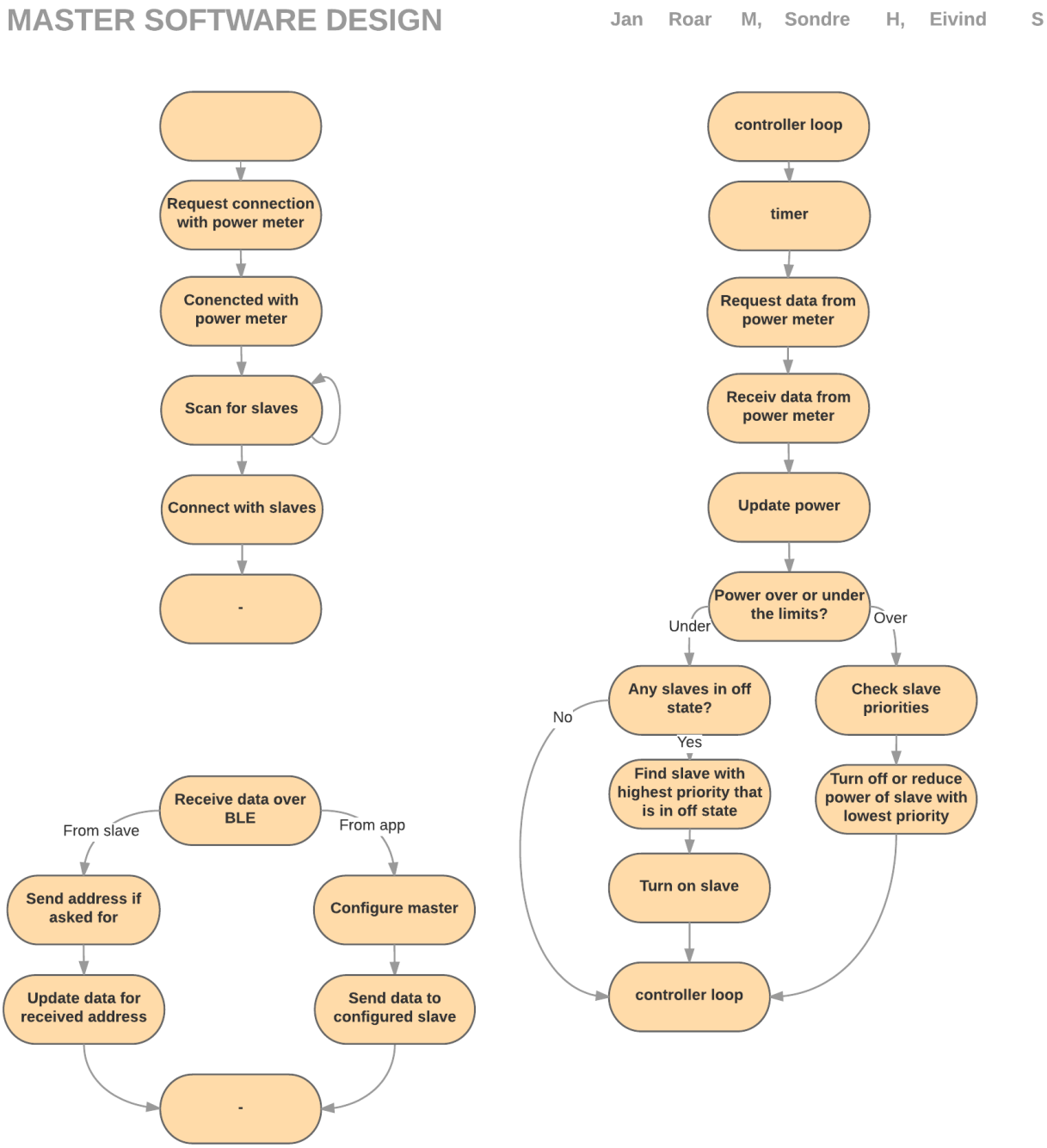


Figure 26: Mind map of how the master software could work

3.2.6 Slave hardware specification

The slave has the ability to read or receive temperature from a device and transmit the values to the master. Based on the information the slave provides the master, the slave receives instructions to turn on or off the device that he is bound to follow.

To turn on/off devices the slave needs to be equipped with a relay that can turn off the devices. A relay is a good solution because it is relatively cheap, easy to implement and have PCB mount possibilities. We have a relay accessible and the coil is galvanic isolated from the switch. This is beneficial in this case, because of the voltage which controls the relay is much lower than the load voltage.

There will be a 12VDC available from a 230VAC to 12VDC transformer connected to the PCB, the 12V is also to be regulated down to 3V3, which is a duplication of the solution created to the master. The relay coil needs 9V to operate the relay switch, also here the same 3V3 regulator will be created with just a slight modification.

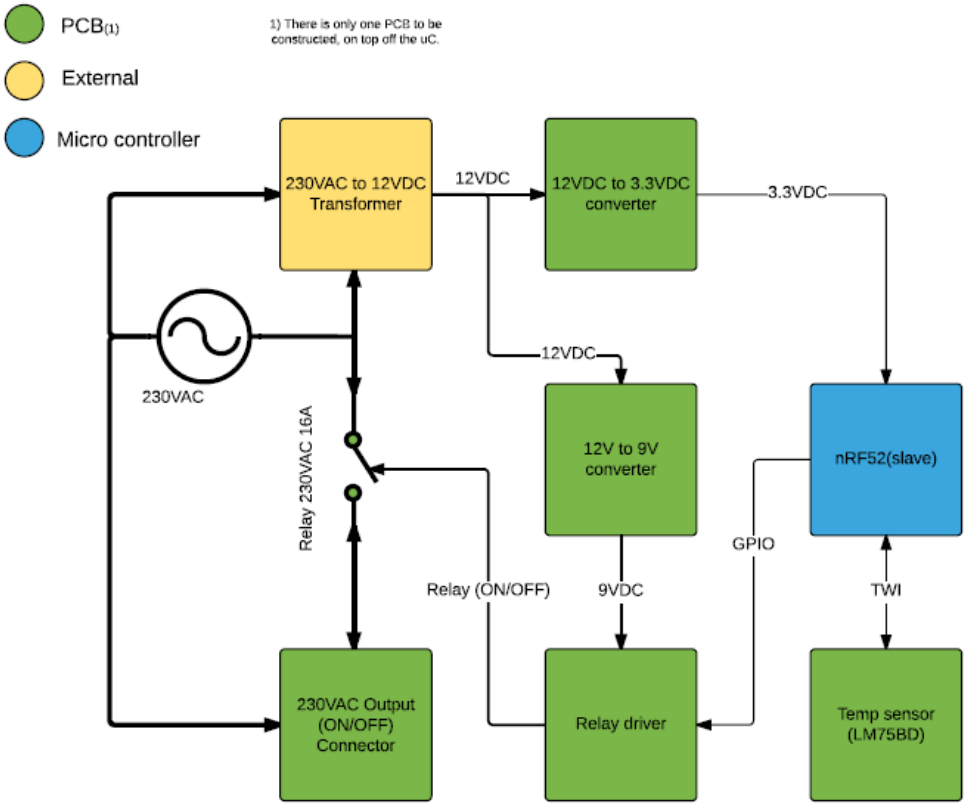


Figure 27: Slave block diagram

3.2.7 Slave software

Figure 28 is an overview diagram on how a slave can be designed. This slave can be controlling an oven, a boiler or electric floor heating, by sensing temperature and controlling a relay as a thermostat. The slave can set his own priority based on the wanted temperature versus current temperate deviation. The master can override the thermostat to turn it off based on the priority set by the slave.

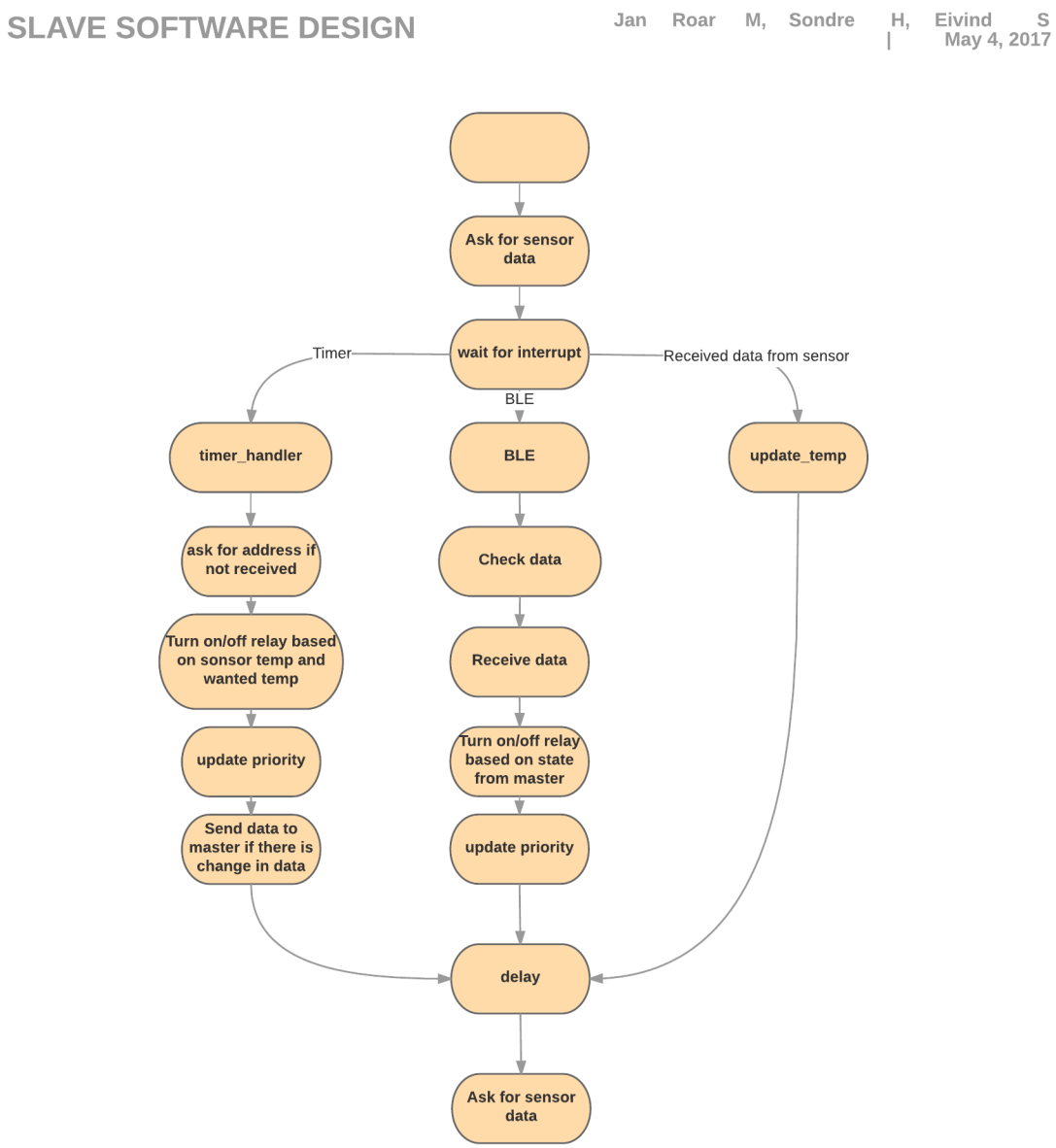


Figure 28: Design specification for the slave device

3.3 Implementation

3.3.1 Code imlementation

BLE code implementation master For the implementation of the master the "Nordic UART Service Client" example is used as template[35]. To communicate with more then one slave the "BLE Multi-link" example[36] is merged with the "UART Service Client" example. On top of this the "Experimental: BLE Relay Example"[37] is merged to add the functionality of letting the master be both a peripheral and a central. This will make it possible for a phone with the nrftoolbox app to connect to the master.

BLE code implementation peripheral For the implementation of the slaves/peripherals, the SDK example "UART/Serial Port Emulation over BLE" is used as template and merged with the Two-Wire-Interface(TWI) example[38][39].

3.3.2 Nordic UART protocol

BLE data and sending Figure 29 shows an overview of the data placement in the datagram, sendt between master and slave devices. The figure 30 shows the dataflow between master and slave.

When sending a datapacket, ack will be set to 0 and an ack timer will be started. If the sender does not receive any packet with ack = 1 before the timer runs out the sender will resend the packet. If the packet gets lost again, it will not try to resend data.

Byte number	0	1	2	3	4	5	6	7	8 -> 20
Description	Node type	Address	ack	state	wanted_temp	Curr_temp	Priority	Emptly	Emptly

Figure 29: Datagram

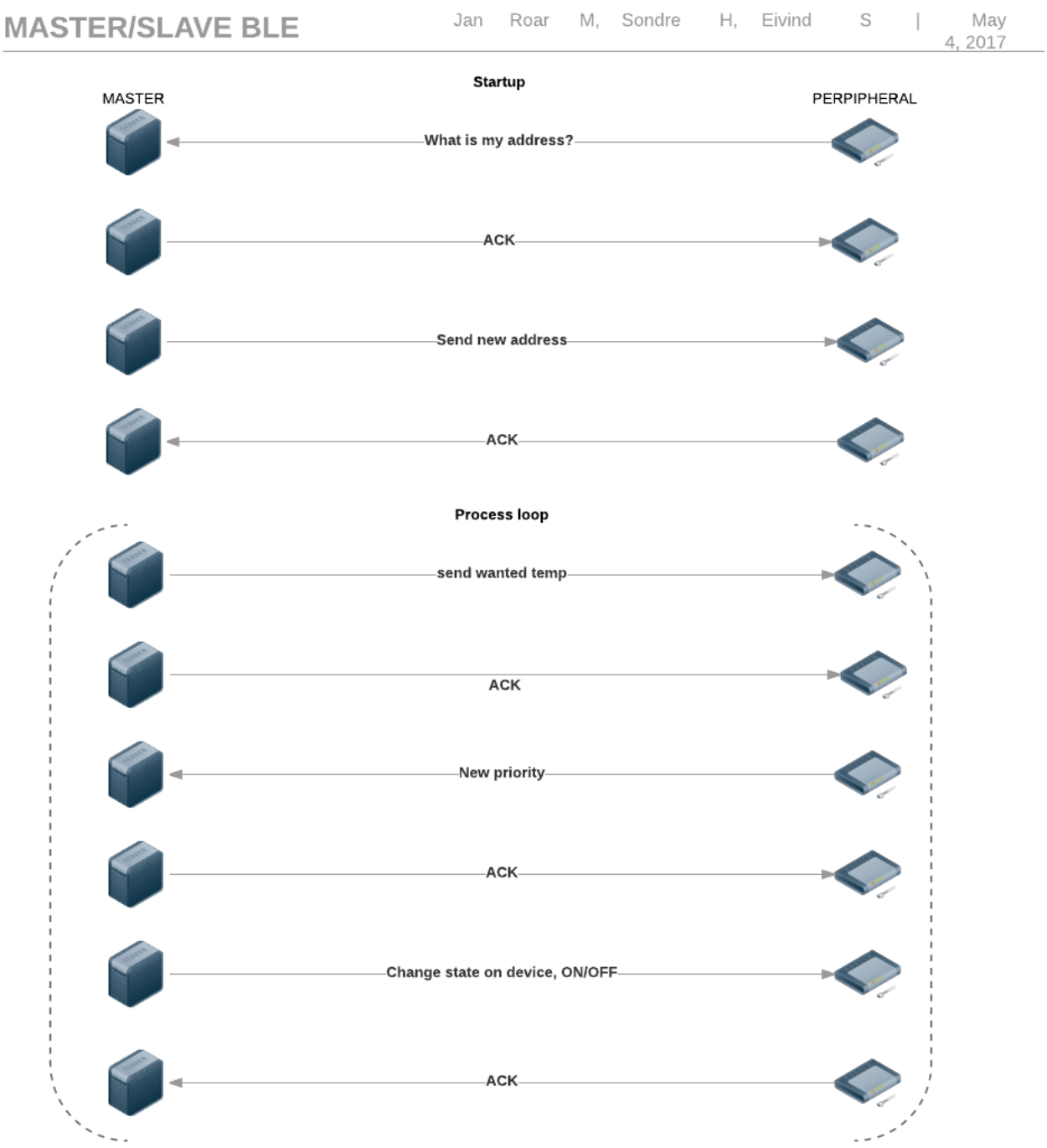


Figure 30: Flowcontrol master/ slave

Master setup The master has to be configured with the nrfToolbox app from Nordic semiconductors[40]. Inside the Toolbox you find a subsection wireless Nordic UART, connect to "El_hub_central" and open the log by clicking on "Show log". You can now send text commands as shown in figure 31.

Power limits

Start by setting the power limits max_consume_limit(h) to the max power allowed by the main fuse in the fuse box of the system. Example: $32A \cdot 230V = 7360W$ -> "limith07368".

Set normal_max_consume_limit to a limit between max and preferred limit. If the consume goes over this limit slaves in normal and low priority will be turned off. Example: "limitn04000".

Set preferred_consume_limit(p) to a limit you want to stay under. If the power companies introduces overprice over a given limit then this is the limit you should use. Example "limitp02000".

Clock

Send in the clock value. Example: "clock0726".

Max power of controlled device

Type "C" slaves has to be set up with a power value, for slave types see Figure 50. This is the max power consume of the controlled device, this can be found at the nameplate. Example: "slave022000".

Wanted temperature

Set wanted temperature on controlled device by slave(n). Example for setting value on slave '1': "temp0120" for 20° C or "temp0120-" for minus 20° C

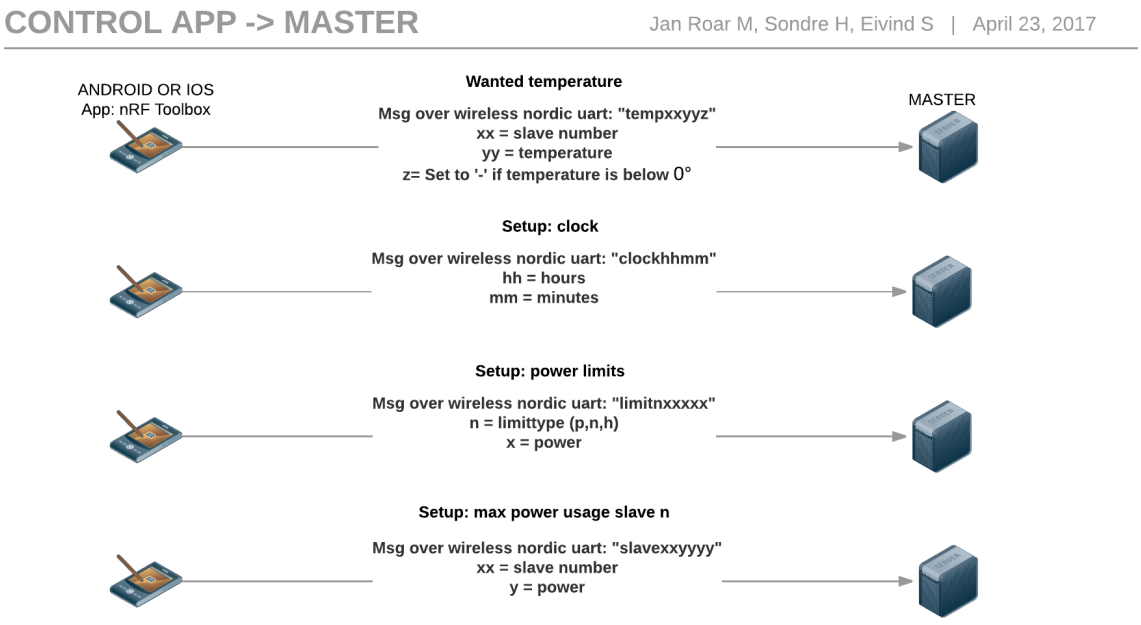


Figure 31: Flowcontrol phone/ master

3.3.3 Priority algorithm

Priority The master will have 3 priority stages, high, medium and low with 10(11) under priorities in each stage, which means there is priorities from 0 to 30 as shown in figure 33. Different devices will have different default priority number and not all devices got all priority levels. For instance boiler got priority 28 in low priority and oven got 25, meaning oven will always have a higher priority in low priority then a boiler. This sub priorities is mainly intended for further development and is not used much in the prototype.

Byte 0/ Types w/temp	w/o temp	
A	a	Master device
B	b	Heat devices w/o boiler
C	c	Boilers
D	n/a	Temp sensor
E	e	etc

Figure 32: Slave types

Overconsumption The algorithm will get information from the AMS meter about the consume. We have invented three limits on the consume:

- preferred_consume_limit:

  limit we want to be under

- normal_max_consume_limit:

  limit we should be under

- max_consume_limit

  limit we can not exceed

This limits has to be set by the user. If we exceed the first limit, the algorithm is starting to order slaves to turn off until the the consume is under the limit or there is no more slaves to turn off. If the consume still rises and we are over normal_max_consume the algorithm starts to turn off devices in normal priorities as well. The same procedure applies for max_consume_limit and high_priority.

Byte 6: Priorities 0->30		Devices
0	HIGH PRIORITY	
1		
2		
3		
4		
5		
6		
7		
8		C
9		
10	NORMAL PRIORITY	
11		
12		
13		B
14		
15		C
16		
17		
18		
19		
20	LOW PRIORITY	
21		
22		
23		
24		
25		B
26		
27		
28		C
29		
30		

Figure 33: Priority distribution

Expensive hours In the close future the power companies will most certainly have changing consume prices through the day, but we do not know how this will be for sure. This is an example on how this can be solved. Since the AMS is not sending the price, the clock is our second option. The algorithm will turn off devices in low priority if the time is between 07:00 -> 10:00 or 17:00->20:00 based on the information in figure 34. The algorithm will not differentiate between weekends (left picture in figure 34) and weekdays (right picture in figure 34), but this should be implemented in a final product.

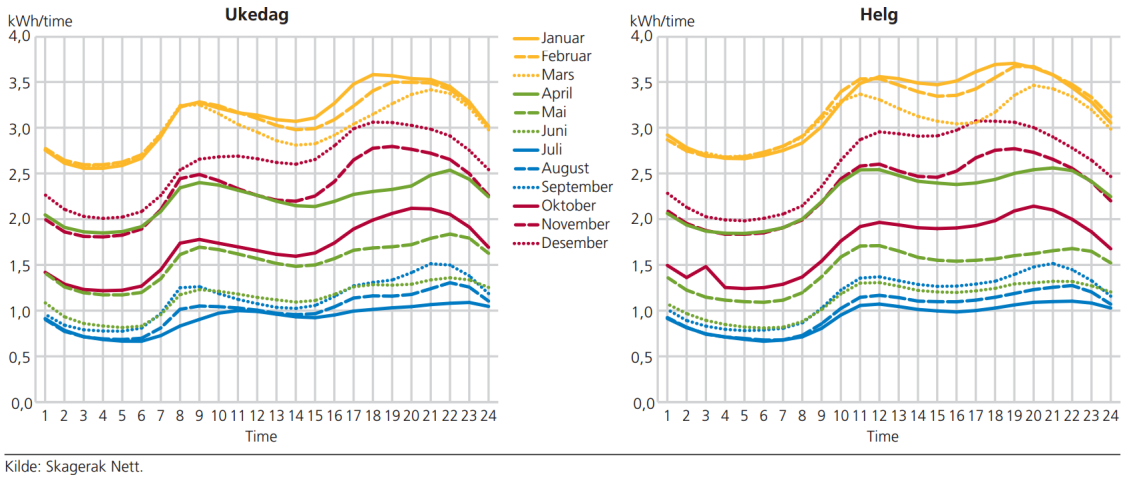


Figure 34: Consumption throughout the day in Norway [3]

Switch on devices When turning off a slave there is a chance that the total consume will go under the consume limit that triggered the controller to turn off the slave. For not having the problem off turning on and off a slave many times in a minute a timer is used on 5 minutes. All slaves share one timer and every time a slave is turned off the timer is started. If the timer runs out the algorithm will first check if the consume is under the preferred_consume_limit, and then search through all slaves and turn on the slave with highest priority that is in off state. If there is more slaves with same priority it will turn off the slave with the biggest temperature deviation. Consequently a slave might stay off for 5 minutes more than it has to.

Led and buttons master

Led 1 blinking: Scanning.

Led 1 on: connected to phone.

Button 2: Start BLE advertising for connection to the nRF Toolbox app.

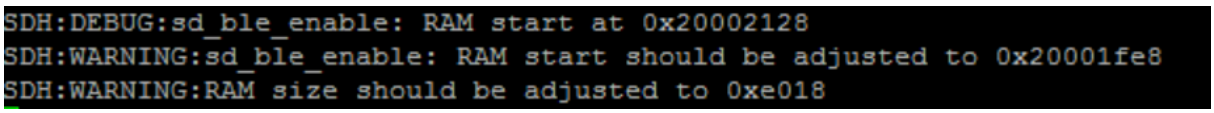


Figure 35: Memory error

3.3.4 Master memory allocation

When implementing FreeRTOS together with BLE and UART the program size started to grow. The program became bigger then the default heap memory allocation for the softdevice and consequently there was delivered a warning. Figure 35 shows one of the warnings, there was several warnings after this. The memory was at last adjusted to the given values in the config file: ble_app_hrs_freertos_gcc_nrf52.ld in: SDK\ examples\ ble_peripheral\ Central\ pca10040\ s132\ armgcc

FreeRTOS total heap size did also needed to be adjusted, in order to have the ability to implement all threads, software timers and sow on. See section 2.8.4 regarding FreeRTOS for more information.

3.3.5 Master block schematic software

This section shows a basic overview of the different threads, binary semaphores, queues, event groups, hardware interrupts and software timers that makes up the software part in the master unit. In appendix 7.6.6 there can be found flowchart of all the different threads, including some of the functions that is called inside some of the threads. More information regarding threads, semaphores, queues, software timers and event groups can be found in section 2.8.4.

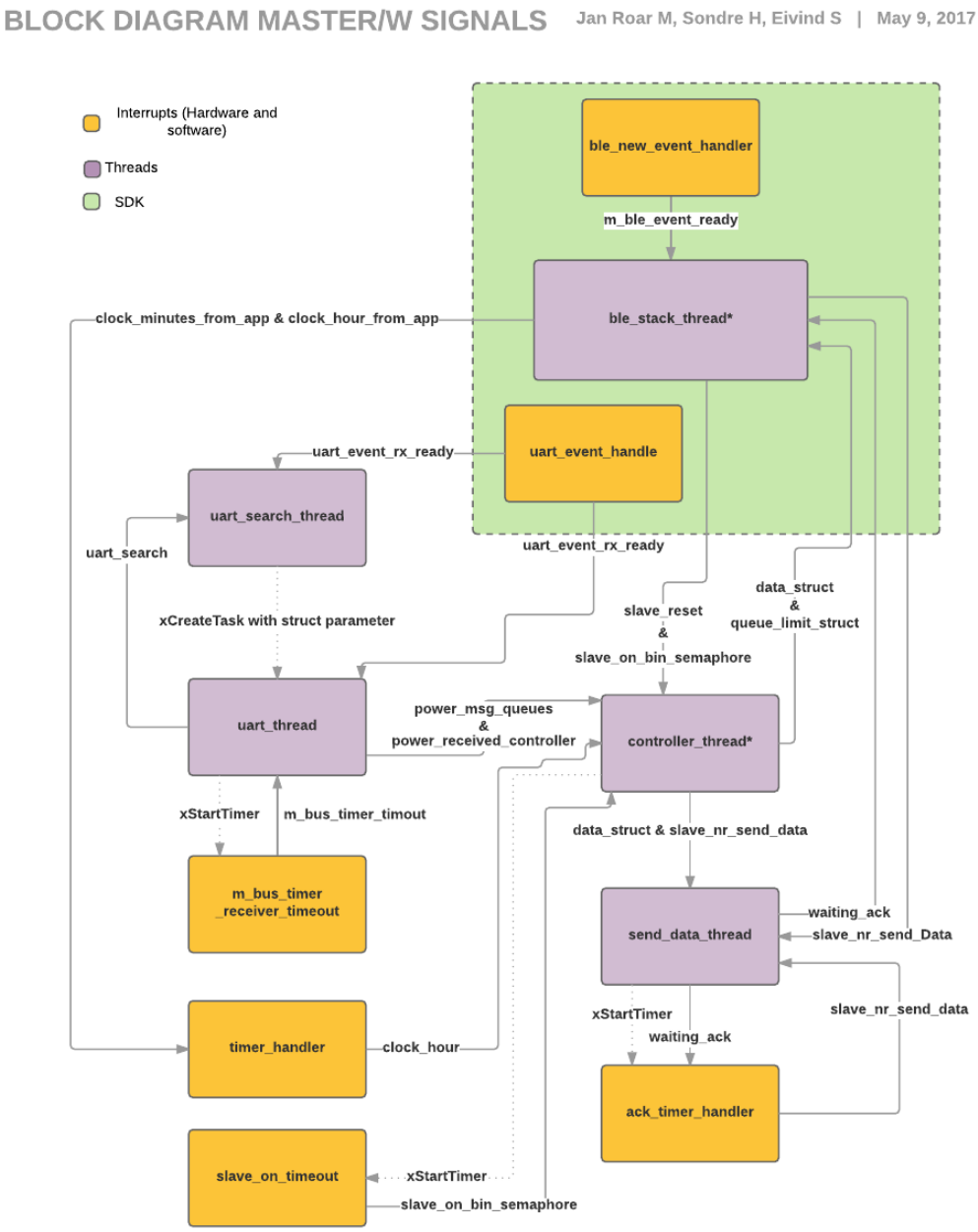


Figure 36: Block schematic software

Binary Semaphores:

- **m_ble_event_ready:** Semaphore raised by ble_new_event_handler when there is a new BLE stack event to be processed in the ble_stack_thread.
- **slave_on_bin_semaphore:** Semaphore raised by either slave_on_timeout when a slave has been off for a given time or by ble_nus_c_evt_handler when a slaves own priority is increased. This semaphores is processed in controller_thread.
- **uart_search:** Semaphore raised by uart_thread if the reading from the m_bus meater goes wrong. This semaphore is processed in uart_search_thread.
- **uart_event_rx_ready:** Semaphore raised by uart_event_handle if there is a new event (Uart data received) to be processed in the uart_search_thread or uart_thread.
- **m_bus_timer_timout:** Semaphore raised by m_bus_receiver_timer when there is time to request a new data from m_bus meater. This semaphore is processed in uart_thread.
- **power_received_controller:** Semaphore raised by uart_thread when there is a new power reading to be processed in the controller_thread.

Queues

- **clock_hour:** This queue is used to send the time (hours) from timer_handler to the controller_thread. This is regarding the Expensive hours found in section 3.3.3.
- **clock_minutes_from_app & clock_hour_from_app:** These two queues is used to set a new time. Sent from nus_data_handler and processed in timer_handler.
- **power_msg_queue:** This queue is used to send the recorded power from uart_thread to the controller_thread.
- **data_struct:** Queue used to hold a struct that points to specific slave structures. This slave structure contains the datagram packet that is sent between the different slaves and the master unit, see section 3.3.2(Nordic UART protocol) for more information. This queue is set by the controller_thread, and there are several function and threads that peeks the queue.
- **slave_nr_send_data:** This queue is received in send_data_thread, and it contains the specific address of a slave. This queues is sent from different threads, functions and software timers.
- **queue_limit_struct:** This queue is used to set the limits sent in from the user, section 3.3.3 (Overconsumption) has a more detailed description of this. This queue is send out in controller_thread, and this queue can be peeked in nus_data_handler.
- **slave_reset:** This queue is sent from on_ble_central_evt to notify the controller_thread about a disconnected slave. The queue contains the address of the disconnected slave.

Event groups

- **waiting_ack:** This event group is used together with ack. More details of this can be found in section 3.3.2 (Nordic UART protocol).

Threads

- **ble_stack_thread** This thread is responsible for handling BLE stack events. This thread is waiting for the m_ble_event_ready semaphore to be raised, after that is calls ble_evt_dispatch functions. More details is found underneath in paragraph 3.3.5.
- **uart_search_thread** This thread is responsible for initialization of the m_bus meter. This thread sends a initialization frame on all available addresses (0-250), if there has been found one or more address then the uart_thread will be started. A more detailed information in form of a flowchart can be found in appendix 7.6.6.

- **uart_thread** This thread will be started if there has been initialized one or more addresses from the uart_search_thread. This thread is responsible for sending request to the m_bus meter, and also read the data from m_bus meter. Inside this thread there is also a check to see if the data from m_bus meter is correctly received. If the m_bus meter don't respond after a certain time, then this thread will delete itself and the uart_search_thread will start over again. A more detailed information in form of a flowchart can be found in appendix 7.6.6.
- **controller_thread** This thread is responsible for the priority algorithm that is described in details in section 3.3.3. A more detailed information in form of a flowchart can be found in appendix 7.6.6.
- **send_data_thread** This thread is responsible for sending data from the master to the slaves. More details of this can be found in section 3.3.2 (Nordic UART protocol). A more detailed information in form of a flowchart can be found in appendix 7.6.6.

Hardware Interrupts

- **ble_new_event_handler:** This is a event handler for a new BLE event. This function is called from the SoftDevice handler, and it is called from an interrupt level.
- **uart_event_handle:** This is the event handler for the UART module. This functions is called when an event occurs in the UART module.

Software timers

- **m_bus_timer_receiver_timeout:** Software timer used to request data from m_bus meter after a certain time.
- **slave_on_timeout:** Software timer used to tell that a slave has been off for a certain of time.
- **ack_timer_handler:** Software timer used together with the ack. More details of this can be found in section 3.3.2 (Nordic UART protocol).
- **timer_handler:** Software timer used to to keep track of time. timer_handler callback function is called every second.

Inside ble_stack_thread Figure 37 found underneath here is just to show where the different semaphores and queues related to ble_stack_thread is located.

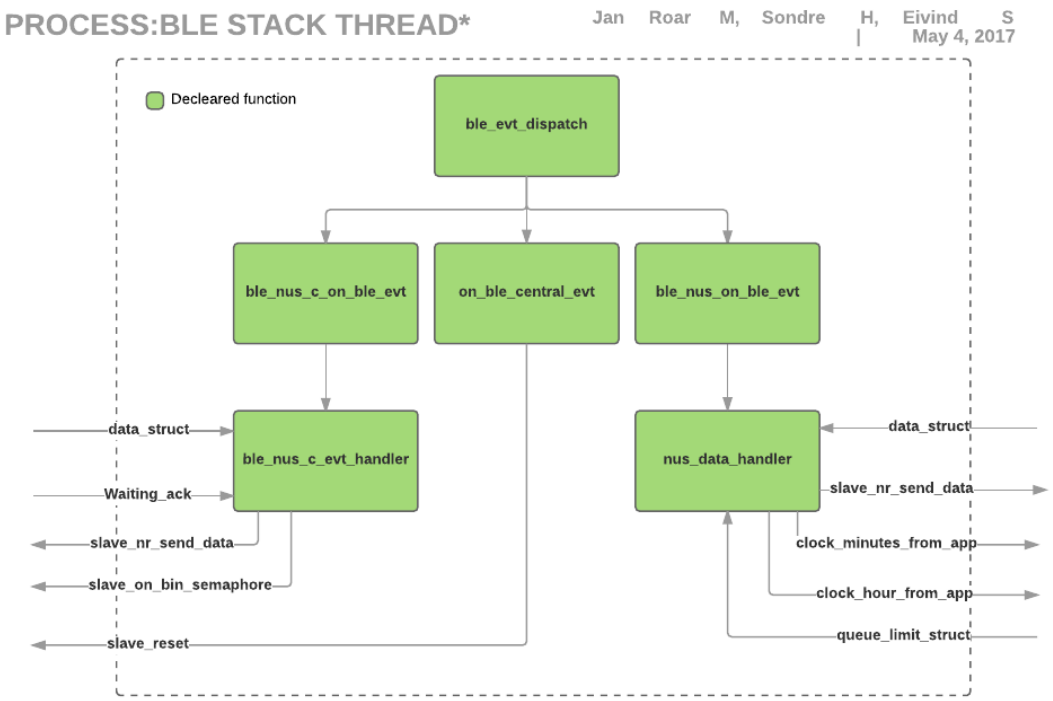


Figure 37: Process: ble_stack_thread

- **ble_evt_dispatch:** This function is responsible for dispatching a BLE stack event to all modules with a BLE stack event handler.
- **ble_nus_c_evt_handler:** This function is called when there is a BLE stack event related to data sent from a slave to the master unit. More information of this data can be found in section 3.3.2 (Nordic UART protocol). A more detailed information in form of a flowchart can be found in appendix 7.6.6.
- **on_ble_central_evt:** This function is called when a slave gets disconnected from the master.
- **nus_data_handler:** This functions is called when there is a BLE stack event related to data sent from control app. More details of this can be found in section 3.3.2 (Master setup). A more detailed information in form of a flowchart can be found in appendix 7.6.6.

3.3.6 Master hardware

230VAC to 12VDC transformer The 230VAC to 12VDC transformer is necessary for the system to be independent and encapsulated, without the need for an external power supply. There will be little time to make one by our own. In order to complete everything, it was therefore decided that it would be used a wall adapter like the one in picture 38 that will be customized to this project.

Figure 38: 230VAC to 12VDC Wall adapter [41]

12V to 3.3V converter The master receives 12V from the transformer, this voltage will be regulated down by a known component LM317, which is a widely used three terminal adjustable regulator from Texas Instrument [42].

nRF52 is a 3.3V micro controller(μC) and therefor it will be constructed a 12V to 3.3V regulator, also to be duplicated and used in the slave design. The circuit have its origin from the data-sheet [42] and modified to fit this project. The circuit is shown in the appendix under master schematics [7.6.3]. In order to get a stable 3V3 output, the Voltage between OUT(Output) and ADJ(Adjust) pin will be held at 1.25V constant, since there is an internal Zener diode, see figure 39, resulting in the desirable output will be 1.25V higher than V_{adj} equation 2.

$$V_{out} = V_{adj} + 1.25V \tag{2}$$

$$V_{adj} = \frac{V_{out}}{1.25V} = \frac{3V3}{1.25} \approx 2.4V \tag{3}$$

This is solved by constructing a voltage divider between Out and ADJ as shown by the resistors R1 and R2 in figure 40b or schematic [7.6.3]. The voltage across R1 will always be 1.25V so the rest of the voltage lies over R2, because of the zener diode internally in LM317, see figure 39.

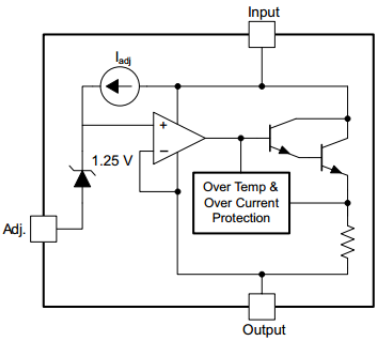
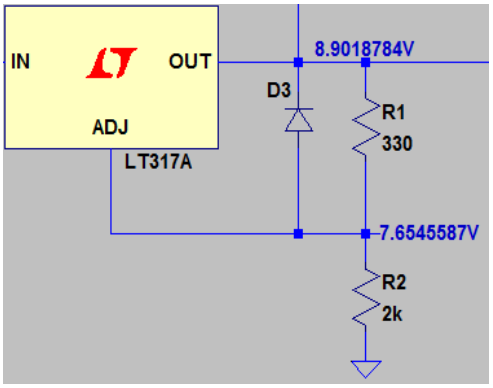
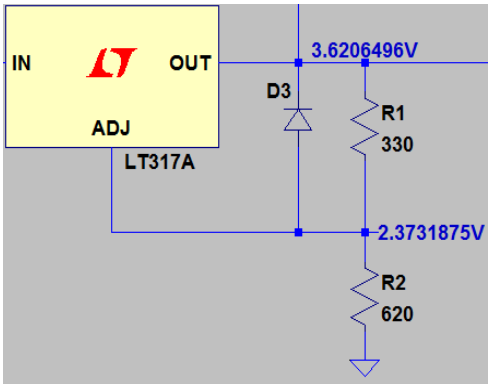


Figure 39: LM317 structure [42]



(a) LM317 9V voltage divider



(b) LM317 3V3 voltage divider

Figure 40: Regulator pictures

Capacitor C3 is placed in the start for filtering any irregularities the supply may produce. There is also placed two capacitors C1 and C2 at the output, since capacitor C2 has a lower resonant frequency than C1, these two capacitors filter a wider ranges of noise than if they were standing alone. As we can see in figure 41, the (black) and (red) line together, forms the typical frequency output characteristics for these two capacitors.

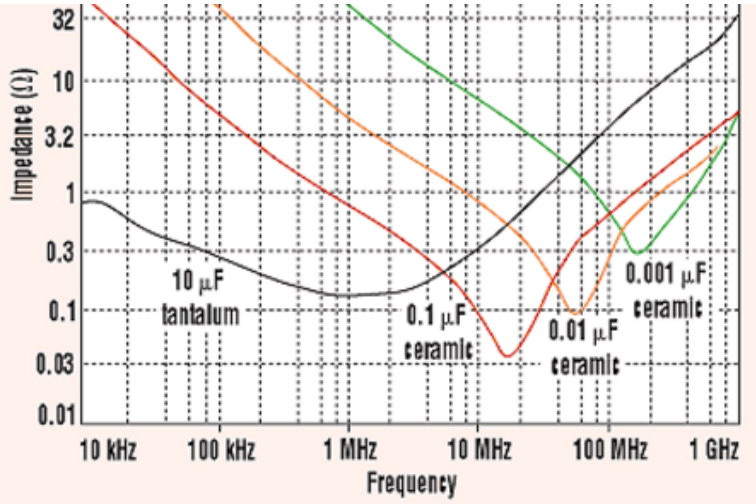


Figure 41: Capacitor filtering capabilities [43]

Wattmeter w/M-BUS interface This watt meter from Saia Burgess[10] with M-BUS interface is going to be used as a replacement for the AMS smart meter. It will deliver the instantaneous power consumption whenever the master asks for measuring values it responds. This should be good enough to imitate the functionality to AMS meter This meter is connected to the simple M-BUS transceiver 3.3.6 through the Meter-bus protocol. More information about the watt meter can be found in section 2.3.

Figure 42: Wattmeter w/M-BUS interface [10]

Simple M-BUS transceiver When the watt-meter from Saia Burgess is to be used, there are several factors to be considered. AMS meter used the M-BUS to supply the slaves with power, which the watt-meter does not. In order to get it to work, there needs to be an alternative way to supply the watt meter, it must also be another circuit that can transmit and then receive information over the M-BUS, because the TSS721a either supports powering of the bus and it might not work with the AMS later on, because it has not been tested against the AMS yet [8].

It was found an example of a M-BUS transceiver on GitHub, made by an user named *RSCADA*. This M-BUS transceiver transmits and receives by using UART, with a separate buck boost converter that generates 34V [44]. There will be used a different kind of buck boost converter, but the transceiver will be used with some slight modifications. The circuit shown in in the appendix 7.5.7 worked well by simulation, results shown in figure 43a and figure 43b, it also worked great under testing, the lab report from the testing found in the appendix at section 7.5.1.

$$\text{R8} = R8.1 + R8.2$$

$$\text{R9} = R9.1 + R9.2$$

$$\text{MBUS+} = 32V$$

R10 and R11.2 voltage divider:

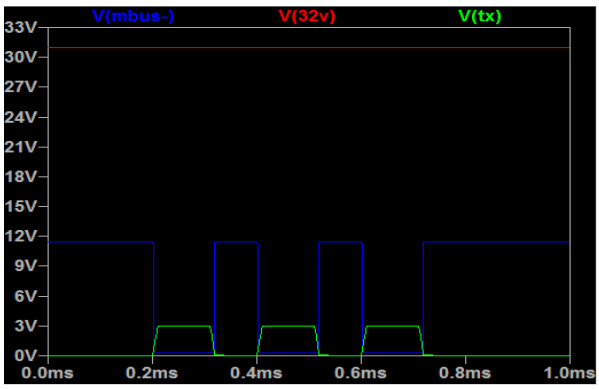
$$V_{rx} = \frac{(MBUS+) \cdot R11.2}{R10 + R11.2} \approx 3V \tag{4}$$

R8 and R9 voltage divider:

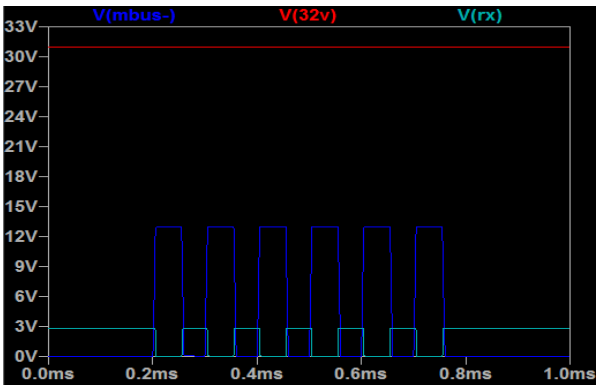
$$V_{T3base} = \frac{(MBUS+) \cdot R9}{R8 + R9} = 12V \tag{5}$$

Transmitting When nRF52 Transmits data over UART to the Tx pin in the circuit 7.5.7 on the simple M-BUS transceiver figure 43a. The voltage generated when transmitting opens transistor T1 pulling down the voltage at the divider R8 and R9 (see equation 5) causes T3 to open because, the voltage at the base is pulled down from 12V to ground, resulting in $MBus- = 0V$ transmitting a logical "one" on the M-BUS.

Receiving When receiving the signal on MBUS- the Transistor T3 opens slightly causing a voltage to appear on base of transistor T2 which opens it and pulls the 3V (see equation 4) down towards ground, causing a signal on the Rx pin.



(a) Simple M-BUS transceiver transmitting



(b) Simple M-BUS transceiver receiving

Simple M-BUS powering In order to get the simple M-BUS transceiver to work the supply wire (MBUS+) needs powering. The PCB 12V available on the PCB, which can be used to power the bus indirectly. The 12V can also be boosted to a voltage fit for powering the M-BUS, optimally the bus power should deliver 36V, that should power the transceiver and the watt-meter M-BUS circuit. The optimal M-BUS voltage is 36V but it is not defining anywhere in the watt-meter data-sheet but, it should work fine with voltages much lower than that. It is then decided to lower the output to 34V to have some margin if there should be some irregularities.

The circuit are familiar for us from an earlier stage, and is bases on LT1072 from Linear Technology, shown in figure 44, which is a high efficiency switching regulator with theoretical output off 65V [45]. The difference of the original circuit in figure 44 is that the circuit is to be modified to an additional voltage of 34V output which is going to fit that purpose. The schematics can be found in section 7.5.7.

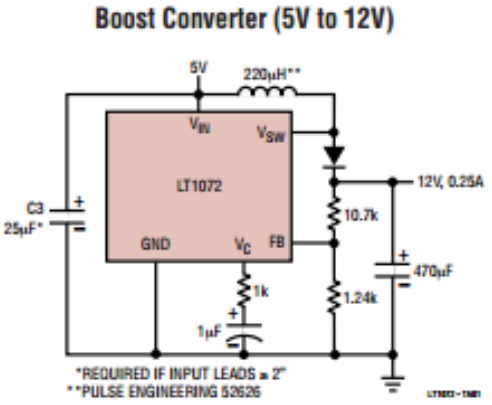


Figure 44: The origin of the buck boost converter
[45]

The Boosted voltage is set by setting the right feedback, V_{FB} needs 1.24V when the right output occur in circuit 45, see equation 6, and the simulated resistor values at figure 45.

The inductor that is used is a little too large, but the components was available in the lab $L = 150\mu F$. C1 is raised from $25\mu F$ to $47\mu F$ since the current input voltage is raised from 5V to 12V, The maximal ripple current, in equation 3.3.6. Ripple current is 145mA by simulating, from equation 9, which gives a RMS value of the ripple current through the inductor 102.5mA 10, making $P_{in} = 1.23W$ equation 11. The output efficiency is determined by the output voltage and the load, which is the resistor of the transceiver which is about $22k\Omega$ and the feedback divider in parallel $P_{out} = 0.09W$ from equation 12, ideally the load resistor should be much which would result in a better efficiency. In figure 46a the inductor ripple current shown in Blue, also known as the input ripple current since it directly connected to the input source, see figure 46. The Green line shows the current through the diode, the current change direction when the current through the inductor falls.

$$V_O = \frac{V_{FB} \cdot (R_1 + R_2)}{R_2} = \frac{1.24V \cdot (50k\Omega + 2k\Omega)}{2k\Omega} \approx 33.5V \tag{6}$$

$$toff = \frac{u_i}{u_o \cdot f} = \frac{12}{34V \cdot 40kHz} \approx 8.8\mu s \tag{7}$$

$$\triangle i = \frac{u_o - u_i}{L} \cdot \frac{u_i}{u_o \cdot f} \approx 800mA \tag{8}$$

$$I_{ripple} = 145mA \tag{9}$$

$$I_{INrms} = \frac{145mA}{\sqrt{2}} = 102.5mA \tag{10}$$

$$P_{in} = I_{INrms} \cdot V_{in} = 1.23W \tag{11}$$

$$P_{out} = \frac{V_{out}^2}{(\frac{1}{22k\Omega} + \frac{1}{27k\Omega})^{-1}} = 0.09W \tag{12}$$

$$Ripple_{\%} = 100\% \cdot \frac{V_{ripple}}{V_{out}} = 0.42\% \tag{13}$$

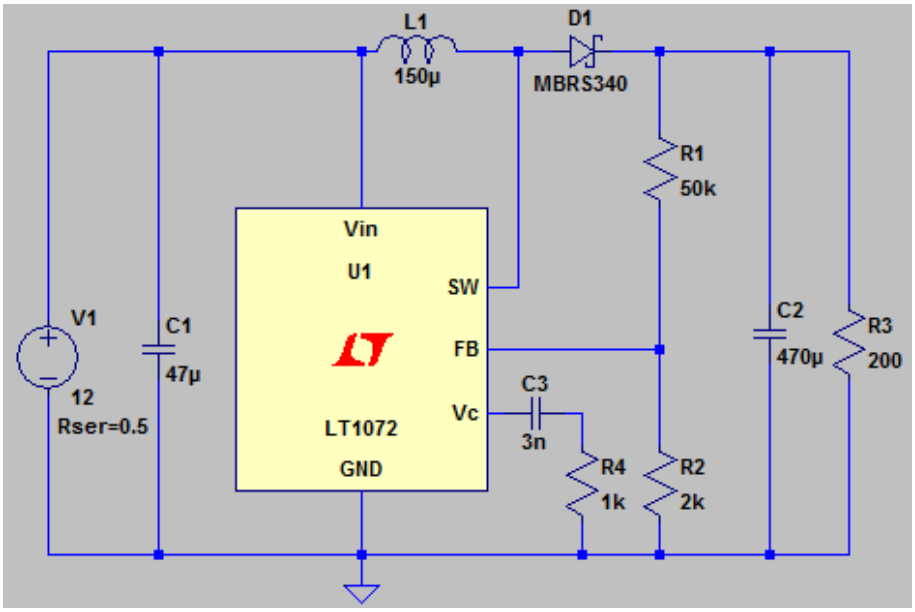
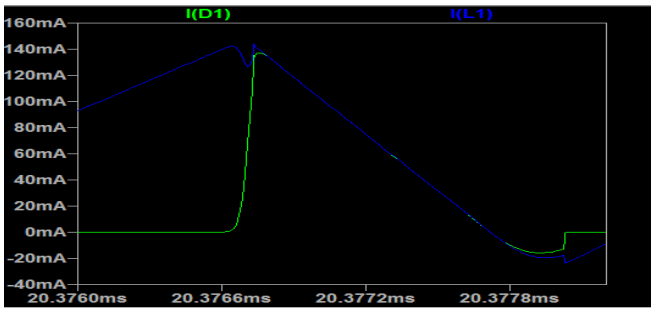
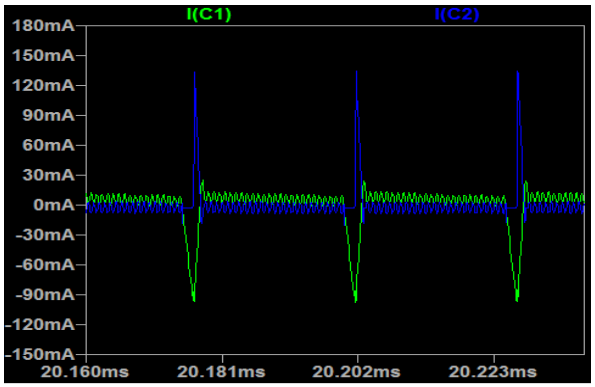


Figure 45: Buck boost simulation circuit



(a) Ripple current inductor and diode



(b) Current input and output capacitor

Figure 46: Simulation results buck boost

The M-BUS transceiver As mentioned in the theoretical background under section 2.2. Developers of the M-BUS protocol suggests a component called TSS721A, that is designed by Texas Instrument [8] that only needs a few external components like resistors and capacitors for operating as a slave device. The slave support a half-duplex communication over M-BUS and convert the information over to UART, which is presented to the μC or from UART to M-BUS sent from the μC. The schematic with the M-BUS transceiver circuit is found in section 7.5.7.

Protective resistors The bus interface consists of two wire, and all slaves connected to the same bus can operate only by the power that the master provides. To avoid danger with short circuit, a resistor should be placed in front of each slave. The bus can reach voltages up to 42V, and we want to limit the current in terms of a short circuit down to 100mA. To do so we need a resistor with size 420Ω equation [9]. Since the bus are polarity independent and for protecting the slave, we split the resistor into two equal resistor in size 220Ω, one on each wire as shown in figure 47.

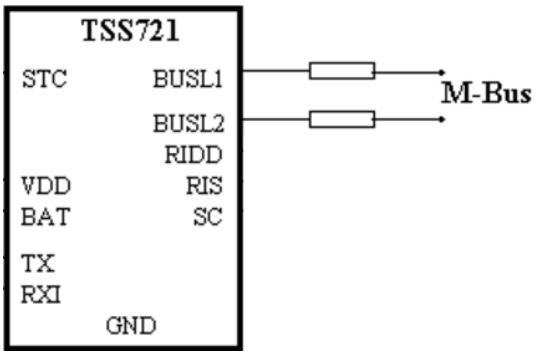


Figure 47: Protective resistors M-BUS [8].

Sampling capacitor When a Master tries to communicate with a slave the slave needs a sampling capacitor C_{sc}, that holds the voltage long enough for the receiver to convert the bits and present it to the μC, see figure 48. There must be sufficient time to recharge the capacitor between the bits arriving. The TSS721A supports baud up to 9600 bps [46], the time spent holding each bit HIGH or LOW is $t_{ps} = 104\mu s$ due to the equation 17. This means the capacitor must be able to charge and discharge within less than $104\mu s$, let's say half to be sure $= 52\mu s$. When receiving a HIGH signal V_{sc} is charged to a voltage of 24V when V_{vb} has reached 36V. When receives a low signal, the bus voltage V_{vb} fall to 24V, and the capacitor C_{sc} starts to discharge when the bus voltage is less or equal the capacitor voltage $V_{sc} = V_{vb}$.

$C_{sc} = V_{vb}$ is separated with a Zener diode shown in figure 48, with a voltage of 12V, and capacitor C_{sc} is charged through this and charged with a rate between 15-30μA, with 36V max from supply and 25 max over the capacitor. Which gives an average resistance about 600kΩ. Calculation for the capacitor size see equation 15.
The size of capacitor $C_{sc} = 178$pF from equation: 15. Its solved with two capacitors we have available in the lab of size 330pF, which results in a 165pF capacitor.

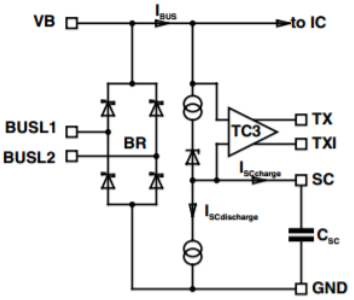


Figure 48: Sampling capacitor C_{sc} M-BUS [8]

Storage capacitor C_{stc} is a storage capacitor which supply the regulated 3.3V output from the TSS721A with power. C_{stc} is charged with a constant current $I_{STC_{use}}$ and is limited by a voltage ref1: 7V. Resistor R_{RIDD} set the charging current of the capacitor C_{stc}, with a value of $R_{RIDD} = 30k\Omega$ gives a charging current = 1mA equation 18, provided by the data sheet [8].

There are only two components connected in series to the VDD a resistor and a capacitor in parallel $R_{vdd} = 100k\Omega$, $C_{vdd} = 100nF$.

Programming resistor When the slave is transmitting information to the master, the bus voltages remain constant, but the slave uses current to transmit the message. The current is unique for each slave and must be programmed by an external programming resistor R_{ris}. This modulation current is recommended to be as large as possible $R_{ris} = 100\Omega$ which gives a 15mA current shown in figure 49, when transmitting over longer distances. The length of the bus in this case will be less than 1m, since this device is most likely to be placed in the fuse box. Therefore the resistor is chosen to: $R_{ris} = 330\Omega$, which result in a programming current equal to $5mA$ in terms of distance and saving power.

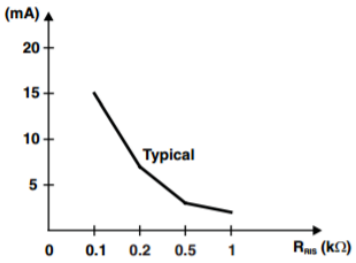


Figure 49: Programming resistor R_{ris} M-BUS [8].

$$C_{sc} = \frac{52\mu s}{600k\Omega \cdot ln(\frac{25}{36})} = 178pF \tag{14}$$

$$R_{zener} = -\frac{36V - 24V}{20\mu A} \approx 600k\Omega \tag{15}$$

$$R_{protect} = \frac{42V}{100mA} = 420\Omega \tag{16}$$

$$t_{ps} = \frac{1}{Baudrate} = \frac{1}{9600} = 104\mu s \tag{17}$$

$$R_{RIDD} = 25\ \frac{V_{RIDD}}{I_{STC}} = 25\ \frac{V_{RIDD}}{I_{STC_{use}} + I_{CS1}} \tag{18}$$

3.3.7 Slave software

For the implementation of the slave devices the nordic UART service template is used as the base of the slave. There will be different slave devices for different purposes.

Byte 0/ Types w/temp	w/o temp	
A	a	Master device
B	b	Heat devices w/o boiler
C	c	Boilers
D	n/a	Temp sensor
E	e	etc

Figure 50: Slave types

Slave types

Boiler:

The boiler slave type 'C' will sense the temperature in the boiler and set the priority value based on the temperature. This data will be sent to the master and the master will respond with a percent number which describes how much power the boiler can use. This code will mainly be done by another group in the project. The priorities the boiler can choose from is LOW(28), MEDIUM(15) and HIGH(8) sends when the priority changes.

Heat devices:

The heat device will sense temperature and regulate as a thermostat based on wanted temperature sent from central. If the temperature is under 2°Clower then wanted temperature, the slave will put the priority to MEDIUM(15), otherwise LOW(25) priority. Sends data when peripherals is not given an address and when change in temperature. It receives a state (0/100) from central which tells the peripheral if it is allowed to turn on. This is typically used on all electrical ovens and underfloor-heating.

Temp sensor:

A slave device with temperature sensor, reports temperature to master.

LED and buttons slave

LED 1: Advertising

LED 2: Not in use

LED 3: State

LED 4: Relay on/off

All LEDs off: Peripheral disconnected and in power mode, needs a restart to reconnect.

3.3.8 Slave code implementation

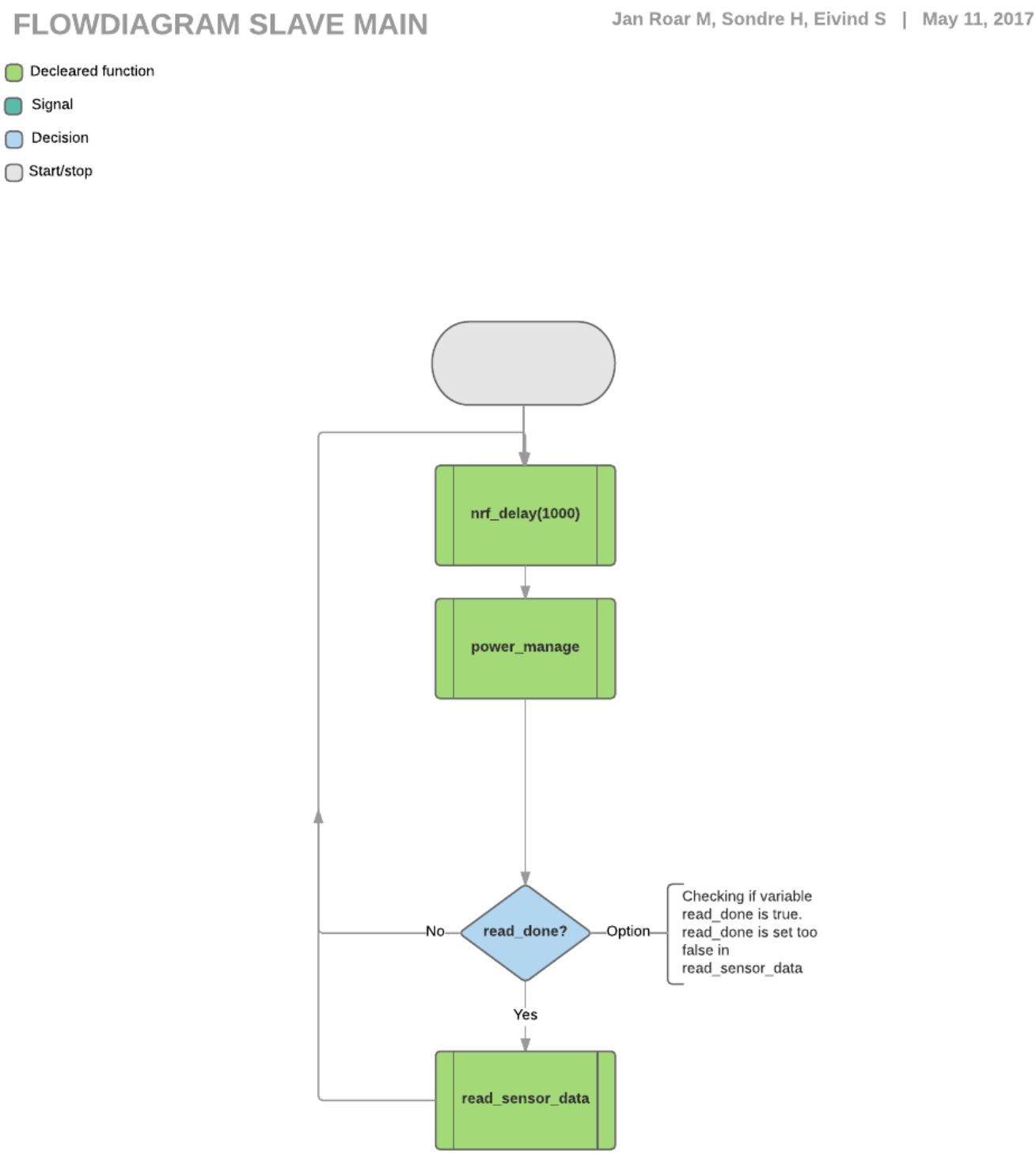


Figure 51: Main function

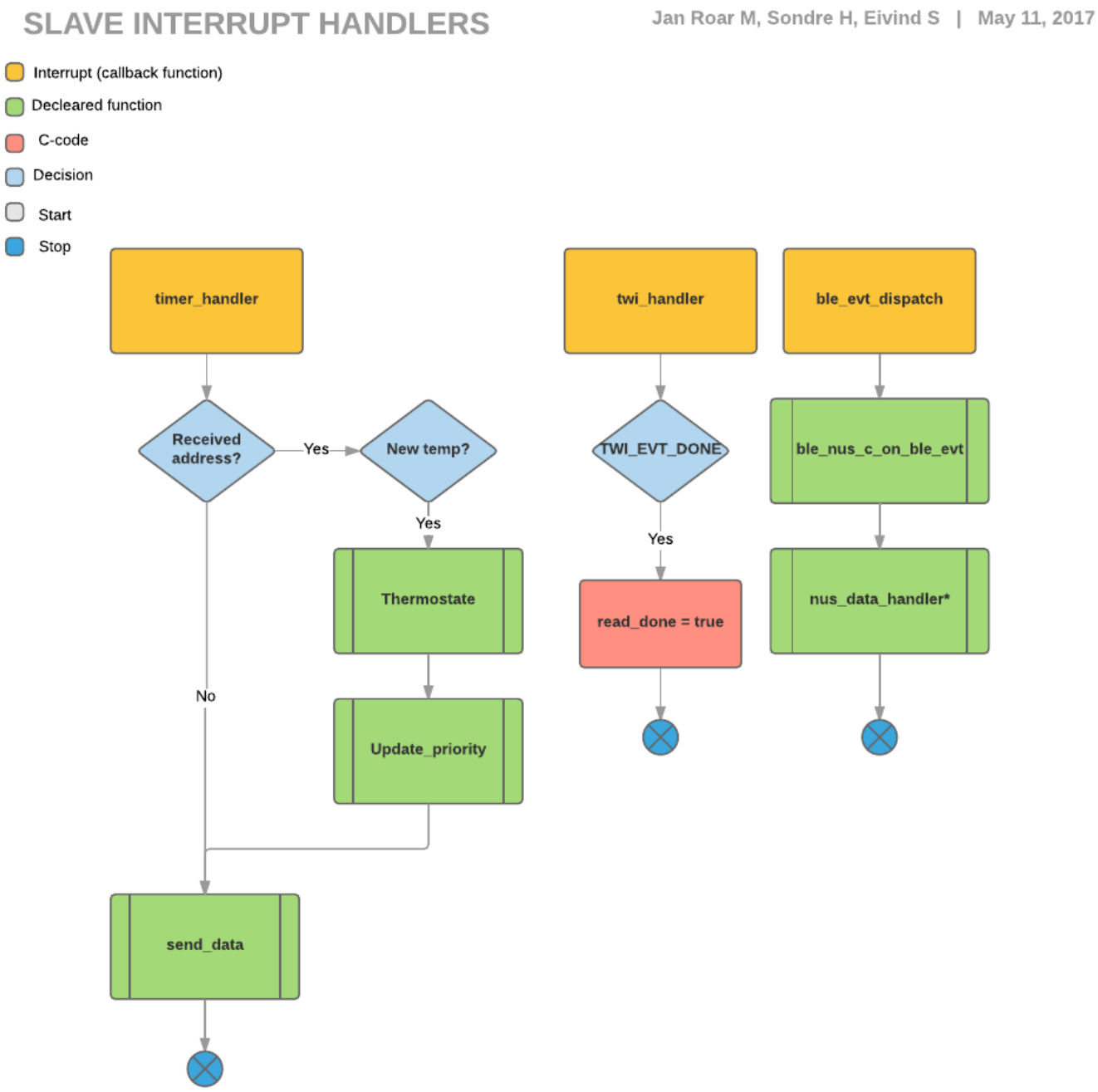


Figure 52: Interrupt handlers

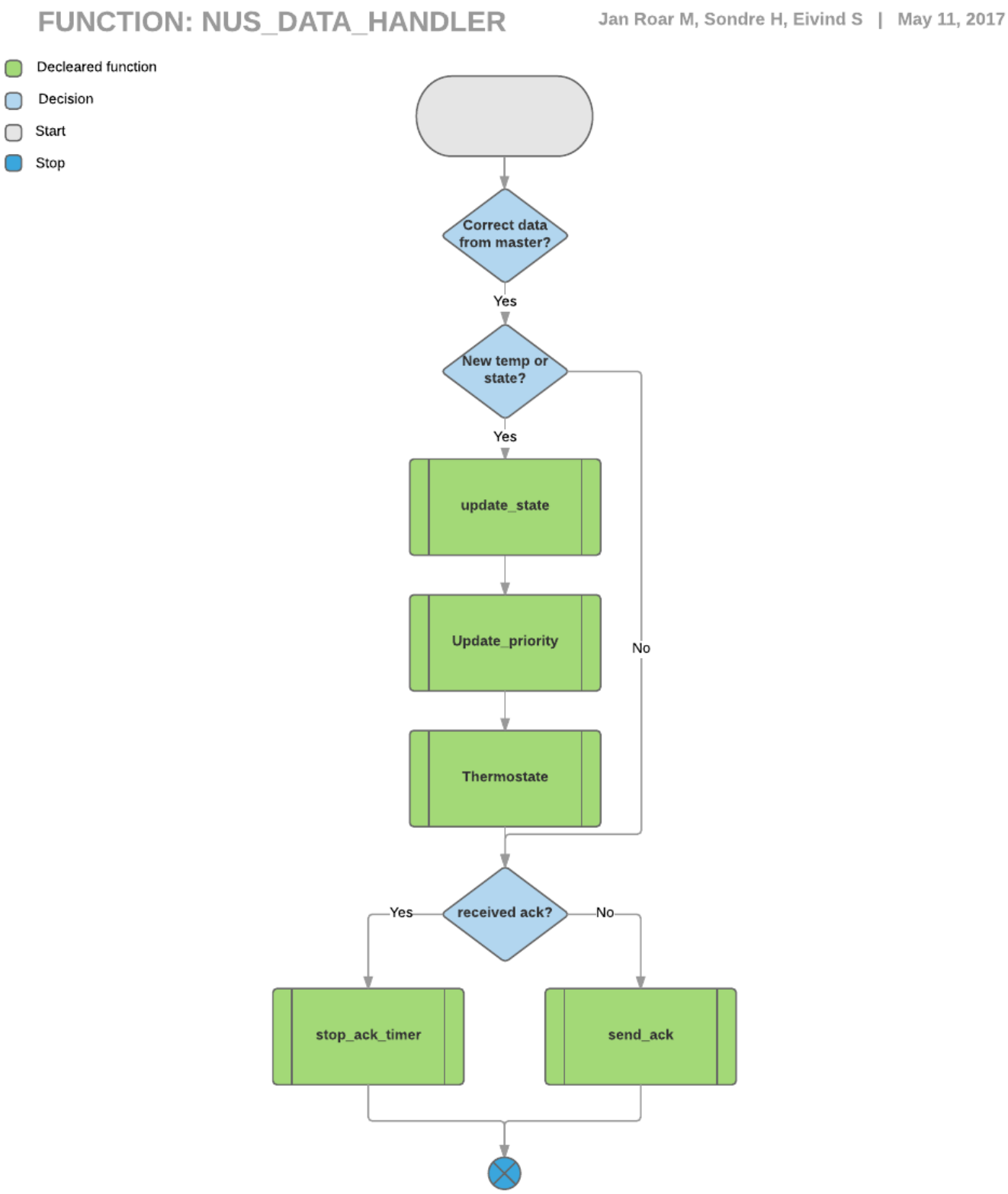


Figure 53: Nus data handler

3.3.9 Slave hardware

Relay The relay that will be used for this project is a HRM2-S DC9V general purpose relay from Multicomp [29] which we will use in normally-open(NO) configuration. It is able to break 230VAC and 16A with a coil voltage about $V_{coil} = 9VDC$ and a coil resistance equal to $R_{coil} = 115\Omega$, which means the coil needs $I_{coil} = 78mA$ to trigger the switch inside, equation 19. It is much current of the circuit ran on battery but it will be powered from the 230VAC and its then considered as insignificant. The relay will get 9V from a regulator, constructed for this purpose.

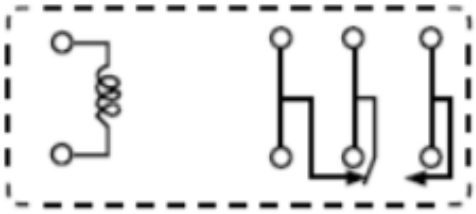


Figure 54: Relay [29]

The relay will be turned on and off from a signal from the nRF52, triggering a pin high or low. The signal is connected to a general purpose transistor in series with the relay which function as a switch, the circuit shown in the schematic located in the appendix 7.6.3 [47]. The circuit is simple, consisting of a general purpose transistor BC548 [48] in series and a flywheel diode in parallel with the relay coil. The transistor creates a switch controlled from the nRF52, its easy to set up and makes it possible to control higher loads with a small voltage. When the transistor breaks the connection from on to off, in a slight moment the relays coils inductance will try to compensate the lac of current when the switch opens and may generate a voltage spike. That is why the flywheel diode is in parallel, to even out these spikes which may damage the circuit.

$$I_{coil} = \frac{V_{coil}}{R_{coil}} = \frac{9V}{115\Omega} \approx 78mA \tag{19}$$

Temperature Slaves are provided with a temperature sensor of type LM75BD [49], which is a temperature-to-digital-converter from NXP Semiconductors. The temperature sensor has a resolution of $0.125°C$ and an accuracy of $\pm 2°C$. It can be supplied directly from the micro controller and uses Inter-Integrated-Circuit(I2C) as communication interface.

There is a small circuit surrounding the LM75BD [49], consisting of pull up resistor for the TWI bus and a 0Ω resistor in order to have a easy way to disconnect the component and a capacitor to filter the irregularities on the power trace. The values of resistor R3, R5, R6 in the circuit 7.6.3 are recommended in the data sheet and is just a percussion, but will not be mounted because the nRF52 got internal pull up resistors. TWI buses SCL, SDA and OS is directly connected to the μC.

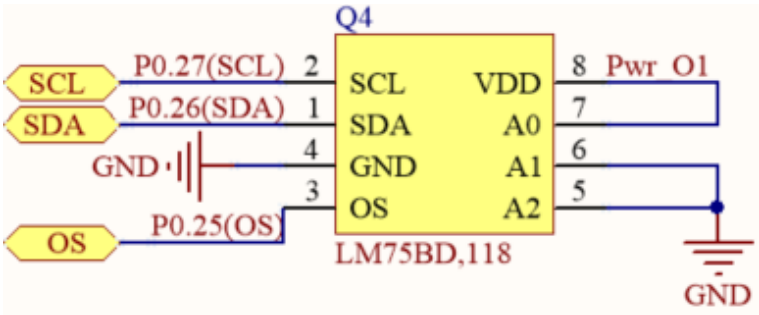


Figure 55: Temp sensor

The surface mounted temperature sensor is not optimally located, ideally it should be placed away from the circuit board because of the copper layer of the PCB collect heat from the micro controller and the circuit around, causing the temperature to raise. On this projects circuit board, a recess was placed underneath the temperature sensor with the same size as the component to reduce this affect. A better solution on previous work would be to place the temperature sensor externally to the circuit board, to get the air temperature but will work great to shown the principle.

3V3 regulator The slave receives 12V from the transformer, this voltage will be regulated down by a known component LM317, which is a widely used three terminal adjustable regulator from Texas Instrument [42].

nRF52 is a 3.3V μC and therefor the slave needs to convert 12V to 3V3, also The relay needs 9V to trigger the relay, 3v3 regulator is already created in the master design, and will be duplicated it in this design to. The 3v3 regulator will also be adjusted to a 9V regulator to use in the relay. The circuit have its origin from the data-sheet [42] and modified to fit this project. The circuit is shown in the appendix under master schematics 7.6.3. In order to get a stable 3V3 output, the Voltage between OUT(Output) and ADJ(adjust) pin will be held at 1.25V constant, see figure 56, resulting in the desirable output will be 1.25V higher than V_{adj} equation 20.

This is solved by constructing a voltage divider between Out and ADJ as shown by the resistors R1 and R2 in figure 57b or schematic 7.6.3. The voltage across R1 will always be 1.25V so the rest of the voltage lies over R2, because of the zener diode internally in LM317, see figure 56.

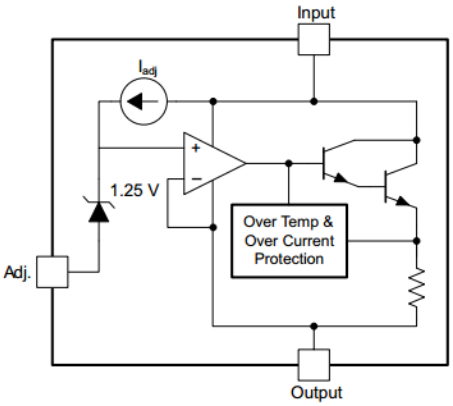
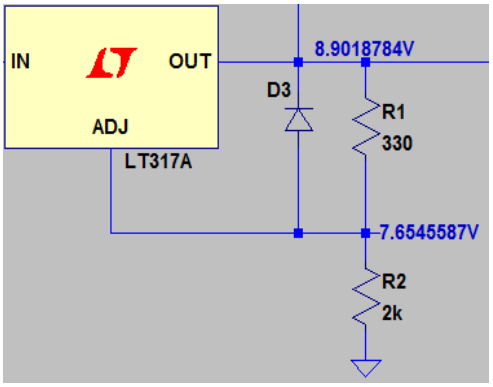


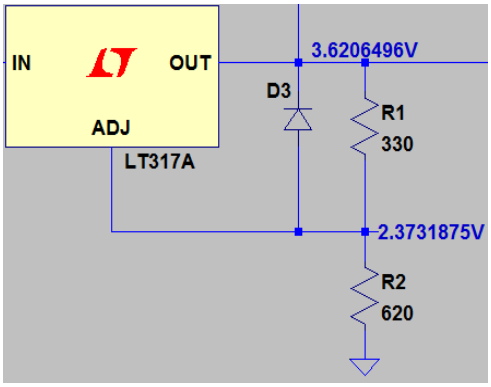
Figure 56: LM317 structure [42]

$$V_{out} = V_{adj} + 1.25V \tag{20}$$

$$V_{adj} = \frac{V_{out}}{1.25V} = \frac{3V3}{1.25} \approx 2.4V \tag{21}$$



(a) LM317 9V voltage divider



(b) LM317 3V3 voltage divider

Figure 57: Regulator pictures

Capacitor C3 is placed in the start for filtering any irregularities the supply may produce. There is also placed two capacitors C1 and C2 at the output, since capacitor C2 has a lower resonant frequency than C1, these two capacitors filters a wider ranges of noise than if they were standing alone. As we can see in figure 41 the (black) and (red) line together, forms the typical frequency output characteristics for these two capacitors.

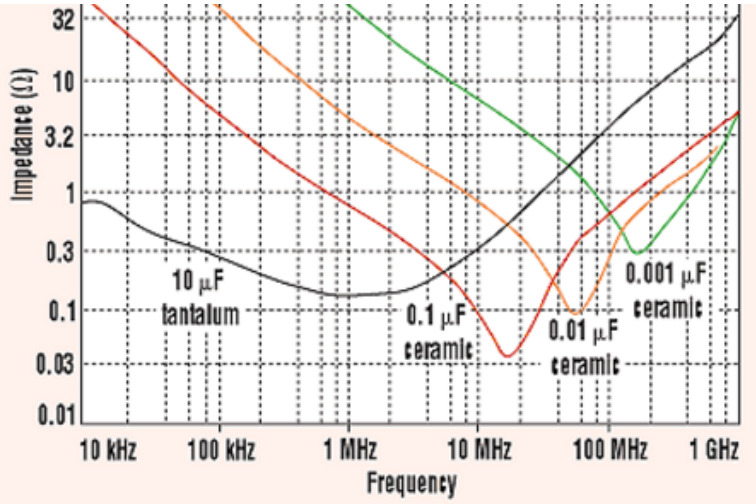


Figure 58: Capacitor filtering capabilities [43]

9V regulator The 9V regulator is based on the same circuit as 3v3 regulator, see schematic in the appendix at 7.6.3. The difference is a slightly change in the feedback resistor R2 is changed to two resistors in series (R20+R30) and increased from 620Ωto 2kΩincreasing the voltage output to 9V see figure 57a.

Both 3v3 and 9v regulator have been tested, test rapport is located in the section 7.5.2.

3.4 Validation & Testing

The original plan was to receive the instantaneous power from the AMS smart meter, but since the firmware was not to be released before in late of May, it stopped the project. The solution was to replace the AMS with a much simpler watt-meter with the same M-BUS-interface and which could provide instantaneous power, in order for the project to go on. The plan was to create two PCBs formed like an "Arduino Shield", which fit's into the nRF52 as well. Every part of the design was simulated and created individually on a breadboard or a PCB. After the construction each circuit got through a comprehensive testing before merged into the master or slave design. The benefit by doing this is that it makes it easy to duplicate these circuits that we know works from the testing in both master and slave design, because both use some of the same circuits.

The master design consist of two M-BUS transceivers, one that communicate with the watt-meter with M-BUS interface and one for the AMS, the AMS transceiver has not been tested, but constructed as instructed from the data sheet[8]. The other transceiver got tested against the watt-meter with a nRF52 to transmit and receive the message, the test report is to be found in section 7.5.1. The design got a buck boost converter and a 12V to 3V3 regulator as well. The Buck boost converter "boosts" 12V to about 34V which are being used to power the M-BUS, test report in section 7.5.4. The 3V3 regulator is for making the master dependent on the 230V to 12V transformer which will be available for the master. The 3V3 regulator was made out off a familiar circuit made out off an LM317 and got through tested, test report: 7.5.2. After all parts were put together the system were tested together to ensure the functionality, against the watt meter and the nRF52 function together, the test report to be found under section 7.5.5.

The slave design is also a single PCB which consist of a duplication of the 3V3 regulator and a 9V regulator which is an adjusted version of the 3V3. The PCB contain a relay for controlling heat sources, the relay is controlled by the μC through a transistor witch function as a switch, in series with the relay coil. The regulator was tested before, and the transistor switch is a simple circuit which we knew worked, and tested together with the whole system. The slave test report found under section 7.5.3.

For software testing there was made a set of test cases for both the slave and master device. The test cases for slave can be seen in test report 7.5.7 and for the master in test report 7.5.6. A few mistakes was found under testing and most of them corrected.

Master: The system is working good, however we get a com error at times, that is triggered by the UART/M-BUS thread. It seems like this problem appears when the M-BUS circuit is not powered. But there was not enough time to rectify this. The sending and receiving of UART data from nRF52 to the M-BUS meter was tested with the logic analyzer as seen in test report: 7.5.1.

Slave: All the functions is working good, however the on-board temperature sensor is not optimal because of the heat from the micro controller. For a functional system an external sensor would be necessary. The sending over TWI works fine, but it can seem like the gpio driver strength could be set a little higher or the frequency of the TWI a bit slower in order to get a faster logic high level.

4 Discussion

This project has resulted in a working system which has been tested and well documented. The group has put a lot of effort into the project, trying to create the best result possible and are proud of the outcome. The project has evolved to be a flexible and highly expendable system, and have reached all the requirements listed 3.1.

The original plan was to read the information that the AMS sends, but it was not ready yet and we were forced to look for alternatives. We found a watt-meter with the same interface which would imitate the AMS(M_1), one difference was that the AMS were powering all the meters connected which the watt-meter did not. We created a master which could power the bus and and function as a transceiver. The master successfully received the instantaneous power consumption from the watt-meter w/M-BUS-interface, through the M-BUS transceiver. The master also have the possibility to communicate with 8 devices through BLE(S_1), organized by an algorithm which is of our creation(M_2). This algorithm gives the slaves addresses and contain priorities between them, based on the information it receives(M_3). The master then uses an protocol we created to receive temperatures and ensure the control and communication between slaves(M_4)(S_2).

The protocol that has been developed between master and slave has room for expansion for further development. There has been developed a simple user interface that uses the Nordic Semiconductor toolbox app on a smart phone(S_3). But it would be beneficial if the application would be equipped with a good user interface like a touchscreen which provides a setup that allow the user to set and read temperatures in every room.

The group has developed a system that preforms very well, and is relatively close to a final product, but if we was to do this project once again, we would create a more structured code with less code in the main file and with more code partitioned into ".c" and ".h" files; We would look for some sort of safety measures for protecting the system against intruders and we would replace the development board to a much smaller module that does the same job and would fit this project better. The hardware for AMS communication is placed on the master PCB, but has not been tested. A major extension would be to upgrade the project to the newest soft device which supports 20 peripherals, and develop code that supports more than 8 slaves and to handle the information that the AMS provides and add it into the already existing algorithm.

This project has taught us to work in teams and that each person in the group contributes in different manner. We have learned to split the responsibilities so every member get their own field of responsibility. This project has taught us to have a better structure in both hardware and software, and made us familiar with the tools surrounding PCB layout and Nordic semiconductors development board which we used and has given us a good overview over protocols that used in this project.

5 Conclusion

The problem was to create a device that should reduce power peaks of a household and make the electricity consumption more constant, based on the instantaneous power consumption provided by the AMS smart meter. The group came up with a working solution to the problem, but we got the announcement that the firmware of the AMS was not to be released before the end of May, forced us to look for a replacement, which we found and continued the project.

The system we created receives the instantaneous power consumption and through an algorithm calculate which slaves that could be turned on and off based on the priorities and temperature sent from each slave. After many hours of work the group have come up with a solution which after hours off testing and debugging performed very well. The whole group was goal-oriented all through the end and the calculated hours in the start of the project came very close to the real-time use.

The Smart Hub we have created solves the consumption problem by smoothing out the consumption throughout the day. It has full control of the consumption by controlling the boiler and heat sources around the system in terms of a more constant power consumption without compromising the comfort. There are many automatic system on the market but in contrast to other system available on the market, this system takes consumption into count which others does not.

6 References

[1] Norges vassdrags- og energidirektorat. *Smarte målere (AMS)*. 2016. URL: http://publikasjoner.nve.no/rapport/2016/rapport2016_79.pdf.

[2] Arne Venjum. "Informasjon til kundene via HAN-grensesnittet i AMS-måleren. OBISkoder." In: *18.03* (2016). URL: 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.

[3] Torgeir Ericson and Bente Halvorsen. *Hvordan varierer timeforbruket av strøm i ulike sektorer?* 2008. URL: https://www.ssb.no/a/publikasjoner/pdf/oa_200806/ericson.pdf.

[4] FreeRTOS. *FreeRTOS - Market leading RTOS (Real Time Operating System) for embedded systems with Internet of Things extensions*. 2017. URL: http://www.freertos.org/.

[5] Nordic Semiconductor. *Development tools and Software / nRF52832 / Bluetooth low energy / Products / Home - Ultra Low Power Wireless Solutions from NORDIC SEMICONDUCTOR*. 2016. URL: https://www.nordicsemi.com/eng/Products/Bluetooth-low-energy/nRF52832/Development-tools-and-Software.

[6] Kamstrup. *Kamstrup ams picture*. URL: http://frifagbevegelse.no/image-11.204916.0d8843d25c?size=1024x0.

[7] NÁVRAT Petr. *Description of OBIS code for IEC 62056 standard protocol*. URL: https://www.promotic.eu/en/pmdoc/Subsystems/Comm/PmDrivers/IEC62056_OBIS.htm.

[8] Texas Instruments. *METER-BUS TRANSCEIVER*. 1999. URL: http://www.ti.com/lit/ds/symlink/tss721a.pdf.

[9] Horst Ziegler. *M-Bus Documentation*. 1998. URL: http://www.m-bus.com/mbusdoc/md4.php.

[10] Saia Burgess Controls. *Energy meter Single phase 230 VAC M-Bus, Saia Burgess Controls, ALD1D5FM00A3A00 | Elfa Distrelec Norway*. 2017. URL: https://www.elfadistrelec.no/en/energy-meter-single-phase-230-vac-bus-saia-burgess-controls-ald1d5fm00a3a00/p/30018167?q=*&filter_Category3=Meters&filter_Category4=Energy+Meters%2C+Power+Monitoring+Devices&filter_Buyable=1&filter_Output=M-Bus&page=5&origP.

[11] SAIA BURGESS CONTROLS. *Single Energy meter with M-bus interface*. 2013. URL: https://www.elfadistrelec.no/Web/Downloads/_t/ds/SBC_EnergyMeter-ALD1-M-Bus_eng_tds.pdf?mime=application%2Fpdf.

[12] Robert Keim. *Back to Basics: The Universal Asynchronous Receiver/Transmitter (UART)*. 2016. URL: https://www.allaboutcircuits.com/technical-articles/back-to-basics-the-universal-asynchronous-receiver-transmitter-uart/.

[13] Zigbee alliance. *zigbee PRO with Green Power*. URL: http://www.zigbee.org/zigbee-for-developers/network-specifications/zigbeepro/.

[14] Phil Smith. *Comparing Low-Power Wireless Technologies | DigiKey*. URL: https://www.digikey.com/en/articles/techzone/2011/aug/comparing-low-power-wireless-technologies.

[15] Smarthome. *What is Home Automation?* URL: http://www.smarthome.com/sc-what-is-home-automation.

[16] Z-Wave. *Z-Wave: The Basics*. URL: http://www.z-wave.com/faq.

[17] Steve Hegenderfer. *Get ready for Bluetooth mesh!* URL: https://blog.bluetooth.com/trashed.

[18] Thread group. *WHY WE MADE THREAD*. URL: https://www.threadgroup.org/About.

[19] EH Contributor. *Home Automation Protocols: A Round-Up - Electronic House*. 2016. URL: https://www.electronichouse.com/smart-home/home-automation-protocols-what-technology-is-right-for-you/.

[20] INSTEON. *INSTEON Details Whitepaper: The Details*. 2013. URL: http://cache.insteon.com/pdf/insteondetails.pdf.

[21] Nordic Semiconductors. *nRF52832 / Bluetooth low energy / Products / Home - Ultra Low Power Wireless Solutions from NORDIC SEMICONDUCTOR*. URL: https://www.nordicsemi.com/eng/Products/Bluetooth-low-energy/nRF52832.

[22] Computer Hope. *What is event-driven programming?* 2017. URL: http://www.computerhope.com/jargon/e/event-driven-prog.htm.

[23] Nordic Semiconductor. *Infocenter Example -Nordic Semiconductor*. URL: http://infocenter.nordicsemi.com/index.jsp?topic=%2Fcom.nordic.infocenter.sdk5.v12.0.0%2Fexamples.html&cp=4_0_0_4.

[24] Robin Mitchell. *Finite State Machines & Microcontrollers*. 2016. URL: https://www.allaboutcircuits.com/technical-articles/finite-state-machines-microcontrollers/.

[25] FreeRTOS. *RTOS kernel rapidly switches between tasks*. 2017. URL: http://www.freertos.org/implementation/a00004.html.

[26] FreeRTOS. *Why RTOS and What is RTOS?* 2017. URL: http://www.freertos.org/about-RTOS.html.

[27] FreeRTOS. *FreeRTOS task states and state transitions described*. 2017. URL: http://www.freertos.org/RTOS-task-states.html.

[28] "Mastering the FreeRTOS ™ Real Time Kernel". In: (). URL: http://www.freertos.org/Documentation/161204_Mastering_the_FreeRTOS_Real_Time_Kernel-A_Hands-On_Tutorial_Guide.pdf.

[29] Multicomp. "Multicomp power relay". In: (). URL: http://www.farnell.com/datasheets/1318173.pdf.

[30] Norcid Semiconductor. *nRF52 Series SoC / Products / Home - Ultra Low Power Wireless Solutions from NORDIC SEMICONDUCTOR*. URL: https://www.nordicsemi.com/Products/nRF52-Series-SoC.

[31] Nordic Semiconductor SDK. *Bachelor - Bachelor*. 2016. URL: http://infocenter.nordicsemi.com/index.jsp?topic=%2Fcom.nordic.infocenter.sdk5.v12.2.0%2Findex.html.

[32] samarkh. *Altium UNO shield template*. 2012. URL: https://forum.arduino.cc/index.php?topic=115446.0.

[33] Saleae. *Saleae Logic Pro 8. The logic analyzer you'll love to use*. URL: https://www.saleae.com/.

[34] *SEGGER - The Embedded Experts - J-Link Debug Probes - Real Time Transfer - RTT Viewer*. 2017. URL: https://www.segger.com/jlink-rtt-viewer.html.

[35] Nordic Semiconductor. *Nordic UART Service Client*. 2016. URL: http://infocenter.nordicsemi.com/index.jsp?topic=%2Fcom.nordic.infocenter.sdk5.v12.2.0%2Fble_sdk_app_nus_c.html&cp=4_0_1_4_2_0_2.

[36] Nordic Semiconductor. *BLE Multi-link Example*. 2016. URL: http://infocenter.nordicsemi.com/index.jsp?topic=%2Fcom.nordic.infocenter.sdk5.v12.2.0%2Fble_sdk_app_multilink.html&cp=4_0_1_4_2_0_1.

[37] Nordic Semiconductors. *Experimental: BLE Relay Example*. 2016. URL: http://infocenter.nordicsemi.com/index.jsp?topic=%2Fcom.nordic.infocenter.sdk5.v12.2.0%2Fble_sdk_app_rscs_relay.html&cp=4_0_1_4_2_1_1.

[38] Nordic Semiconductors. *UART/Serial Port Emulation over BLE*. 2016. URL: http://infocenter.nordicsemi.com/index.jsp?topic=%2Fcom.nordic.infocenter.sdk5.v12.2.0%2Fble_sdk_app_nus_eval.html&cp=4_0_1_4_2_2_18.

[39] Nordic Semiconductor. *TWI Sensor Example*. 2016. URL: https://infocenter.nordicsemi.com/index.jsp?topic=%2Fcom.nordic.infocenter.sdk51.v10.0.0%2Ftwi_sensor_example.html.

[40] Nordics Semiconductors. *nRF Toolbox for BLE – Android-apper på Google Play*. 2017. URL: https://play.google.com/store/apps/details?id=no.nordicsemi.android.nrftoolbox.

[41] ROHS. *Wall adapter 230VAC to 12 VDC*. URL: https://sc02.alicdn.com/kf/HTB1UB2XMVXXXXcWapXXq6xXFXXXZ/Us-au-eu-uk-12v-wall-mounted.jpg.

[42] Slvs044x –. "LM317 3-Terminal Adjustable Regulator". In: (1997). URL: http://www.ti.com/lit/ds/symlink/lm317.pdf.

[43] ANDY aka. *Where did the value of 0.1uF for bypass capacitors come from? - Electrical Engineering Stack Exchange*. 2015. URL: https://electronics.stackexchange.com/questions/172447/where-did-the-value-of-0-1uf-for-bypass-capacitors-come-from.

[44] RSCADA. *M-Bus tranceiver github*. URL: https://github.com/rscada/libmbus.

[45] Linear technology. *LT1072 - 1.25A High Efficiency Switching Regulator - Linear Technology*. URL: http://www.linear.com/product/LT1072.

[46] Sparkfun. *Serial Communication - learn.sparkfun.com*. URL: https://learn.sparkfun.com/tutorials/serial-communication/rules-of-serial.

[47] Electronics Tutorials. *Relay Switch Circuit and Relay Switching Circuit*. URL: http://www.electronics-tutorials.ws/blog/relay-switch-circuit.html.

[48] Fairchild Semiconductors. "BC548". In: (). URL: http://pdf.datasheetcatalog.com/datasheet/fairchild/BC548.pdf.

[49] NXP Semiconductors. "LM75B Digital temperature sensor and thermal watchdog". In: *Rev. 6.1* 6 February (2015). URL: http://www.nxp.com/documents/data_sheet/LM75B.pdf.

7 Appendices

7.1 Appendix A - Abbreviations & Glossary

Dictionary	
Abbreviation	**Meaning**
6LoWPAN	IPv6 over Low power Wireless Personal Area Networks
AC	Alternating Current
ACK/ack	Acknowledge
AMS	Advanced Measuring system
ARM	Semiconductor and software design company
BLE	Bluetooth Low Energy
bps/BD	bits per second/BAUD RATE
CPU	Central Processing Unit
DC	Direct Current
FSM	Finite state machine
HAN	Home Automation Network
I2C	Inter Integrated Circuit
ID	Identification
IDR	Integrated Development Environment
IEEE	Institute of Electrical and Electronics Engineers
IP	Internet Protocol
KNX	Name on a smart house solution
LED	Light Emitting Diode
LSB	Least Significant Bit
M-BUS	Meter Bus protocol
MOSFET	Metal–Oxide–Semiconductor Field-Effect Transistor
NC	Normally closed
NO	Normally open
NOK	Norwegian Kroner
nRF52	Nordic semiconductors SoC microcontroller
NVE	Norwegian Water Resource and Energy Directorate
OBIS	Object Identification System
PCB	Printed Circuit Board
RAM	Random Access Memory
RMS	Root Mean Square
RTOS	Real Time Operating System
RX	Receive
SDK	Software Development Kit
TTL	Transistor-transistor logic
TWI	Two Wire interface
TX	Transmit
UDP	User Diagram Protocol
UiA	University of Agder
UART	Universal Asynchronous Receiver/Transmitter
UPD	Universal powerline bus
WIFI	Wireless Local Area Networking
w/ temp	With Temperature
w/o temp	Without Temperature
μC	Micro-Controller

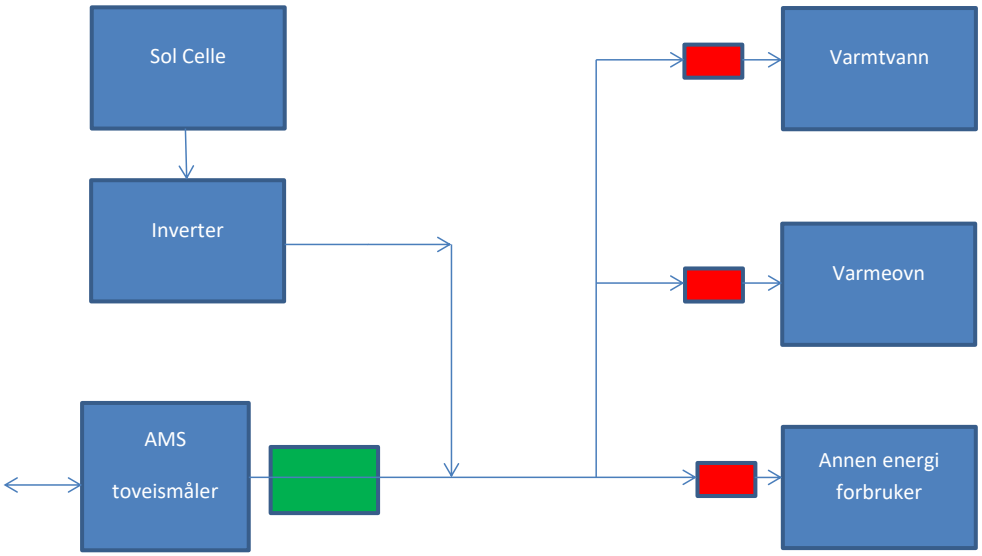
7.2 List of Figures

List of Figures

7.3 Appendix B - Meeting/Gantt diagram/Timesheets

7.3.1 Given Task

3 Bacheloroppgaver Elektronikk
Smart Strøm



Smart effekt styrer: (Smartsikring)

1. Vet hvor mye effekt som er tilgjengelig
2. Vet hvor mye effekt energi forbrukerne ønsker
3. Vet hvilken prioritet effekt forbrukerne har.
4. Kan bestemme hvilene energi forbrukere som får forbruke effekt. (og hvor mye)

Smart effekt kontroller:

1. Kan styre effekten som forbruker får. Av/PÅ eller trinnløst.
2. Vet hvor mye effekt som energi forbrukeren ønsker. Sender denne informasjonen til Smart effekt styrer.
3. Vet hvilken prioritet energi forbrukeren har, sender denne informasjonen til Smart effekt styrer.
4. Vet om energi forbrukeren kan ha trinnløs effekt styring og sender denne informasjonen til, Smart effekt styrer.

7.3.2 Guidance meetings

1st Guidance meeting

Date: 27.January 201
Who was at the meeting:
Group 8: Sondre, Jan Roar and Eivind.
Group 5: Adrian Langemyr, Henriette T. W. Grønlund, Wesley Ryan E. B. Paintsil.
Supervisor: Geir Jevne.

Report:
We discussed what protocol to use to communicate with our nodes. Geir suggested Obis. We need priorities and sub priorities, a program that is easy to expand.

2nd Guidance meeting

Date: 13.February 2017
Who was at the meeting:
Group 8: Sondre, Jan Roar and Eivind.
Supervisor: Geir Jevne.

Report:

- The flow chart was complex and it is difficult to see how things are connected, especially Register Device (block that does not have arrow into it). Geir proposes a solution that has multiple scenarios, one based on ammeter and based on the M-BUS. It makes the chart easier.
- He proposes to turn together ADC processes a block, maybe enough with ammeter block instead of ADC.
- Look for standards and protocols that exist in the market, and may possibly come with improvements, to name a few protocols and compare.
- Writing about Ammeter solution (downside) and compare the use of 2 or 3 current transformers by measuring three phase.
- Writing about the use of setback, the regulation of housing.
- Reading up on fuses and reaction, to see how often we need to read the ammeter to regulate effect.

3nd Guidance meeting

Date: 15.Mars 2017
Who was at the meeting:
Group 8: Sondre, Jan Roar and Eivind.
Supervisor: Geir Jevne, Ken Henry Andersen.

Report:

- We need to specify the section about MOSFETS better.
- We need to put colors into the flow-diagram and specify it.
- We need to look into timer1 to see if it is reserved.
- Change name on slave device-manager
- Event loop should be written about as a code implementation.
- Write about what people have done/ main responsibility.
- We need to find out if our M-BUS slave got the power supply integrated.

4th Guidance meeting

Date: 19.April 2017
Who was at the meeting:
Group 8: Sondre, Jan Roar and Eivind.
Supervisor: Geir Jevne, Ken Henry Andersen.

Report:

- Jan Roar asked the supervisors if the flow diagram over the software is OK.
- Jan Roar should avoid shortcuts in the code to make it easier to read and debug.
- Supervisors likes the hardware block diagram over the slave, but need a block for temperature sensor.
- A small presentation about the algorithm is given to the supervisors. Priority and functionality.
- Give a good reason reason why we should wait 15 min or more before starting to turn on devices (algorithm).
- We don't use much time on the AMS part, just describe what is done and that we did get it in time.
- The watt-meter(Plan B) needs to be described in the theory part.
- The supervisors want the first revision of the report, when its finish to look though it.
- Jan Roar ask if there was some way to run the same code i different threads. There should be a solution.

5th Guidance meeting

Date: 03.May 2017
Who was at the meeting:
Group 8: Sondre, Jan Roar and Eivind.
Supervisor: Geir Jevne, Ken Henry Andersen.

Report:

- Version control, maximum 1 page.
- Preface is OK.
- Use BLE instead of ble, use uppercase letters for shortcuts UPB,BLE and so on.
- First time, write the whole name, and use the shortcut after this.
- We use the name protocol to many places, can we merge this?.
- Remove TWI and write under nRF52 which part we use off the nRF52.
- Change event driven "sits and wait" to "waits".
- Requirements overview, need to separate for wired and wireless com.
- The requirements needs to be numerated.
- Powering of the M-BUS transceiver and BLE needs to be in the BOCK DIAGRAM.
- Software overview: We must remove the ASK part.
- Color on Block diagram, and differ which is signals or power.
- A better solution on the request of temperature in the slave software.
- Differ what happens in main what happens in interrupt contex.
- Change the Bluetooth protocol to "Master to slave protocol".
- ICP protocol, could have been used¿
- Indicate that the process in flow control repeat itself (loop).
- Priority algorithm, check 27 or 28 priority.
- Specify the 15 minutes in switch on device better.
- Update flow diagram and describe it better.
- Use thread or TASK, not both.
- Simple M-BUS transceiver calculation needs pictures
- The M-BUS transceiver, remove first part about feasibility report.
- Merge the BLE functions flow diagram slave device.

6th Guidance meeting

Date: 11.May 2017
Who was at the meeting:
Group 8: Sondre, Jan Roar and Eivind.
Supervisor: Geir Jevne, Ken Henry Andersen.

Report:

- Report outline should be created
- Main goal
- 7,5 hour in work plan and decimals, per week. move coloms,
- 2.2 look for grammar
- 2.3 look at the grammar
- 2.3 describe master and slave, Master provide slaved to the rest
- Baud rate,
- Master slave protocol????? 2.5 confusing
- Finite state machine rewrite "back to the beginning"
- Remove 2.8.5
- functional requirements at least one slave??
- 3.2.1 energy effective standard, is not a requirement??
- Master and slave green (on the pcb) describe it better PCB/micro-controller osv.
- bobble overview of the software in the design specification. rename the figure caption "tankekart over software".
- Design specification for the salve device
- real time clock page 26??? remove real time clock! or describe the RTOS clock better,
- Wanted temperature, describe slave 1 and slave 2.
- FB figure and volt divider can be removed in buck boost
- add a comment to read done.
- slave code implementation? to much space
- Testing and validating, a little more about each testing
- mention AMS in the start of the discussion
- to hard to read thee discussion.
- Need to mention in the conclusion.
- change M-BUS to M-BUS
- Sort the abbreviation alphabetically
- figure in the press release
- name the appendix
- refer to the software diagram in the appendix

7.3.3 Group meetings

1st Meeting

Date: 10.January 2017
Who was at the meeting: Sondre, Jan Roar and Eivind.

Generally We have decided how the project should look like and how we should approach the problem, has also distributed tasks for the next time.

Tasks until next time

Jan Roar:

- doing research on protocol for communication between master and slave.

Sondre:

- Finish report template

Eivind:

- Fesability report- introduction

New meeting Thuesday 10.11.16

2nd Meeting

Date: 17.january 2017
Who was at the meeting: Sondre, Jan Roar and Eivind.

From last time:

Jan Roar:—

Sondre: Finished report template for feasibility study, Bachelor report, and wrote a part of the project description.

Eivind: Worked with project description

Tasks until next time

Sondre and Eivind will meet the supervisor tomorrow and get the project approved.

Jan Roar: research
Sondre: research M-bus

Eivind:

- research

Next meeting Monday 30.01.17

3rd Meeting

Date: 30.January 2017
Who was at the meeting: Sondre, Jan Roar and Eivind.

Generally:
Pilot project report was put on standby until the meeting with renewable. The focus was directed to another project we have, which deals partly the same and which indirectly can be used in feasibility report.

Tasks until next time

Jan Roar:

- Working with BLE and FreeRTOS.
- Can we use some example code?
- Writing about FreeRTOS
- NUS (Nordic Uart Service)

Sondre:

- Examining the components we need for the project and what solutions we are going for.
- Starting to draw M-bus receiver in Altium Designer.
- Writing about M-bus receiver and smart power meters in the feasibility report.

Eivind:

- Programming: Working with BT and free RTOS, try to communicate with master and slave. Try to find example code

New meeting Thuesday 09.02.17

4th Meeting

Date: 9.February 2017
Who was at the meeting: Sondre, Jan Roar and Eivind.

From last time:
Sondre: Sondre: Completed issue and the theoretical background on M-bus, Ordered part for the project and started working on M-bus design in Altium Designer.

Jan Roar: Completed reading: "Mastering the FreeRTOS Real Time Kernal A Hands On Tutorial Guide". Fixed problems with compiling large c project. Started compiling inn GCC (Commando line).

Eivind: Been working on BT protocol, configured GCC compiler and worked with SDL diagrams.

Tasks until next time:
We must find out how much time we need for the tasks in the project, until Monday.

Jan Roar:

- Create SDL(Flow) diagram Friday/Monday
- Write more about state machine vs RTOS on the feasibility report, and make a conclusion about RTOS(FreeRTOS)

Sondre:

- Create a template solution for Gantt diagram.
- Create SDL(Flow) diagram Monday.
- Continue working on M-bus Receiver.

Eivind:

- Create SDL(Flow) diagram Monday
- Read about OBIS Code

New meeting Wednesday 15.02.17

5th Meeting

Date: 15.February 2017
Who was at the meeting: Sondre, Jan Roar and Eivind.

From last time:
Sondre: Did finish template for the Gantt diagram, and helped with the flow diagram, also did some work on the M-BUS receiver but not finished because we prioritized to finish the feasibility study.

Jan Roar: Almost finished the flowdiagram of the master unit(need some modification). Started writing about state machine and RTOS, not finished yet.

Eivind: Wrote about protocols and fixed a better system for time sheet.

Tasks until next time:

Jan Roar:

- Rewrite SDL/flowdiagram after the meeting
- SDL/flowdiagram of the slave.
- Describe both of the flowdiagram(master and slave)
- Continue writing about fsm and rtos(freertos)

Sondre:

- Finish writing about ammeters in the master unit.
- Finish the Gantt diagram.
- Write about the extensions.
- Look for other alternatives than OBIS codes.

Eivind:

- Finish work with protocol, obis code and make a better timesheet

New meeting Wednesday 22.02.17

6th Meeting

Date: 22.February 2017
Who was at the meeting: Sondre, Jan Roar and Eivind.

From last time:
Main focus last week, was to complete the feasibility study. The report got completed and delivered on Monday.

Sondre: Made a Gantt diagram but, it got complex and unorganized, so i made a new one in Excel, I wrote about extensions and did some research about obis-codes, also did some work on the Transceiver.

Jan Roar: Finished the flowdiagram of central and slave device, and made a short description of these. Also finished the description of fsm and rtos.

Eivind: I finished writing about the protocol, obis cosem codes and made a better time-sheet. We was finished with the feasibility report on Sunday. So I worked with the SDK BLE protocol this week.

Tasks until next time':

Jan Roar:

- Making a provisional uart for implementing uart on the central. With the help of an Arduino Leonardo
- Setting up the uart init on nrf52
- Obis code from my uncle

Sondre:

- Complete the design of the M-BUS transceiver.
- Make a logic level converter 5V-3.3V, to the UART(just for testing UART).
- Finish writing about M-BUS transceiver in implementation and design, main report.
- Look for a M-BUS sender, for professionalize the AMS.

Eivind:

- Study ble SDK for nrf52
- make a connection with uart between central and a perepheral
- Send NUS data between devices

New meeting Monday 06.03.17

7th Meeting

Date: 06.March 2017
Who was at the meeting: Sondre, Jan Roar and Eivind.

From last time:

Sondre: I finished the design and soldering of the M-BUS transceiver, and its ready for testing and debugging. together with our supervisor we found a replacement for the AMS smart meter, since we will not have the opportunity to work on the AMS device. i did not finish writing about the M-BUS transceiver in the report, because of work on the slave device.

Jan Roar: Didn't finish the uart implementing, but started little bit of it. Made a starting project of the central device with FreeRTOS (implemented 2 examples from nordic into one). Made a demo of uart with a Arduino Leonardo.

Eivind: Worked with the connection between slave and peripheral. Until now I have only managed to connect and send nus data to one slave. I have also startet to set up GITHUB to make it easier for to or more people work on same code.

Tasks until next time:

Jan Roar:

- Setting up the uart init on nrf52, making a new task
- Make a algoritm of some sort on how to send and receive uart, according to the Energy meter Single phase 230 VAC M-Bus, ALD1D5FM00A3A00, Saia Burgess Controls
- Continue with ble and freertos

Sondre:

- Create a slave with temperature sensor.
- See if we need to use ammeter?
- Finish writing about M-bus transceiver.

Eivind:

- GITHUB setup
- Work on protocol between slave and master
- Set up a temp sensor on slave

New meeting Monday 17.03.17

8th Meeting

Date: 17.March 2017
Who was at the meeting: Sondre, Jan Roar and Eivind.

From last time:

Sondre: Created a slave, finished the temperature design but we decided to do some changes to the slave, and create a relay that can be controlled. Since we don't get the AMS, we ordered a watt-meter with M-BUS interface which shows the same principle. We don't need the ammeters anymore. This new watt-meter don't work as expected and we need to power the M-BUS externally. We have also worked on another report, that have been prioritized the last 1.5 weeks. I did not got the chance to finish M-BUS because of problems with the new watt-meter. Ordered some parts to the slave since last time.

Jan Roar: Had problems with the initiliasing of the uart after i implemented the NRF_LOG module. Figured out that the uart module had to be initialised before the logging function. After that i got error in the softdevice, had to change memory addresse start and the size of the memory in a freertos file. The after implementing som more task i got errors, then i had to change the heap size in a freertos config file. After that everything with freertos and softdevice worked fine. Then i continued with the uart implementation, and m_bus implementation. Made a new header and c file regarding the m_bus watt meater. Made a dummy nrf52 that gives the central reply on the uart.

Eivind: I had some problems with the implementation of github but it is now working properly. I have managed to connect the master with 3 slaves over ble. It should work with up to 8 slaves but I have only tried 3. I am not finished with temp sensor on slave. But the code whould be finished by the week.

Tasks until next time:

Jan Roar:

- Continue with the UART implementation/ m_bus.
- Writing about this in the report.

Sondre:

- Find a solution to powering the M-BUS.
- Work on a solution for the slave when the parts arrives.
- If we find a solution for the M-BUS, we need to test the M-BUS receiver i made previous.
- Finish writing about M-bus transceiver.

Eivind:

- Finish temp sensor and slave device
- Report writing slave

New meeting Wednesday 22.03.17

9th Meeting

Date: 22.March 2017
Who was at the meeting: Sondre, Jan Roar and Eivind.

From last time:

Sondre: We found a solution for externally powering the M-BUS. We built a new and simpler transceiver and we manage to connect it to the nRF52 and receive the measurement values.

Jan Roar: Had some issues with the m_bus dummy, figured out that i can test directly on the m_bus. Making a plan on how to read the data from m_bus. Figured that out. Found out that i need to make a flow chart over the uart implementation, before it gets to big. Got connection from m_bus meter when sending request, but not initialization.

Eivind: Been sick and haven't done anything.

Tasks until next time:

Jan Roar:

- continue with uart, making a flow chart.
- decoding the telegram structure from the m_bus receiver, and writing about this i in the main report.

Sondre:

- Send the simple M-BUS transceiver to print.
- Start working a solution for relay 230V 16A on the slave device.
- Write about the simple M-BUS and powering

Eivind:

- Still sick, but I will try to do last weeks goal if I get recovered.

New meeting Monday 27.03.17

10th Meeting

Date: 27.March 2017
Who was at the meeting: Sondre, Jan Roar and Eivind.

From last time:

Sondre: I have been sick since Thursday, and haven't done anything.

Jan Roar: Startet with a flow chart, found out that giving a mutex in an isr is not allowed. Had to implement this another way. Made codes for sending initialisation, reset, changing primary address and reset partial power on the m_bus. Discussed with Eivind the algoritm of the central device. We found out that the slave has to be a thermostat, or else it wount work as planned.

Eivind: Wrote about the slave in report, slave is working with temp sensor, implemented ack in the sensing.

Tasks until next time:

Jan Roar:

- Finishing the uart
- Figuring out initialisation of m_bus.
- Flow chart finish
- Writing in the report.

Sondre:

- Write about the simple M-BUS and powering.
- Start working a solution for relay 230V 16A on the slave device.
- Solder the simple M-BUS transceiver.

Eivind:

- Write about protocol in the report
- Start on master logarithm

New meeting Friday 12.04.17

11th Meeting

Date: 12.April 2017
Who was at the meeting: Sondre, Jan Roar and Eivind.

From last time:

Sondre: I have printed slave devices and soldered them, i tested the relay and regulators and it worked well. I have also been working from home a few days and written on the report and worked on the Master design. I have been tested the buck boost converter for powering the M-BUS before sending it to print.

Eivind: I have written about the protocol in the report and made a master logarithm without FreeRtos. The code is working but I have started to change the code to implement FreeRtos.

Jan Roar: Thought i was finished with the uart, but apparently i figured out new and bether way to fix it. Figuring out how to "find" m_bus slaves. Making a task that sends request to the respective slave devices (only one now). Sow have to make a new flow chart, or just fix the one i already had made.

Tasks until next time:

Jan Roar:

- Finish the UART code
- Finish the flow chart
- Start with some writing in the report

Sondre:

- Get a working buck boost converter and integrate it into master design.
- Complete/solder the master until next meeting ready for testing.
- Complete the slave, fix the powering fault.
- Complete test/lab reports.

Eivind:

- I will try to fix the code with freeRtos.

New meeting Wednesday 19.04.17

12th Meeting

Date: 19.April 2017
Who was at the meeting: Sondre, Jan Roar and Eivind.

From last time:

Sondre: I have been working on the Master, but i need to figure out the Buck boost converter before i can complete the design, i have also worked on the report.

Eivind:The code is working with freeRTOS

Jan Roar: Finished the uart code (or so i hope), but it is not tested yet. Flow chart over uart is almost close to finish, just some small details. Started writing in the theory part of the report.

Tasks until next time:

Jan Roar:

- Checking the slave code, and figured out the temp sensor.
- Testing the central device.
- Flowchart diagram of controller task
- Report: Writing about M-Bus central
- Report: Writing about Uart
- Report: Writing about Rtos and Freertos

Sondre:

- Test and complete the buck boost converter so the master design can be sent to print.
- Write about the buck boost converter on the theoretical part of the report.
- Change some details on the HW BLOCK DIAGRAMS
- Solder the Master design after its finish from print.

Eivind:

- Merge main code with updated version of UART
- Fix references with mendeley and IEEE standar
- Report: Controller
- Make it possible to adjust max power of slave and clock with app

New meeting Monday 24.04.17

13th Meeting

Date: 24.April 2017
Who was at the meeting: Sondre, Jan Roar and Eivind.

From last time:

Sondre: Soldered the master and its now ready for testing on Wednesday. I have been working on the report, included the schematic and made a better outline. Written lab reports.

Eivind: Merged code with uart, fixed references, added functionality to phone -> master BLE protocol

Jan Roar: Not finish with all my task, because the meeting was sooner than planned. Finished the uart flow chart. Started a overview flowchart, with all the "signal" that are sent from task to task. Starting figuring out Eivinds code, since i shall make flowchart for that too.

Tasks until next time:

Jan Roar:

- Finish flowchart
- Writing about the flowchart in the report
- Writing in the theory part.

Sondre:

- Test and complete the Master design.
- Find box to the slave and mount a socket.
- Find another 12V adapter.
- Do a system test when everything is complete.
- Finish writing about Buck boost converter.
- Complete M-BUS in the report.
- Write function requirements.

Eivind:

- System test, add functionality to phone-> master protocol (feedback to phone commands),start on preface if there is time

New meeting Wednesday 03.05.17

14th Meeting

Date: 03.May 2017
Who was at the meeting: Sondre, Jan Roar and Eivind.

From last time:

Sondre: Got to finish the hardware and we tested the system. I have been writing a lot on the report and made some changes. Didn't find a box for the slave, that will be handeled later on.

Eivind: Finished with SW testing, but not test report. Finished the phone master protocol code. Wrote the preface and made flow charts to slave.

Jan Roar: Have not finished the flowchart completely, and therefore i did not started writing about the flowchart in the report. Did started with the theory part, just missing a tiny bit there.

Tasks until next time:

Jan Roar:

- Flowchart must be finish
- Finish theory part
- Make a test report about Uart
- start writing in design specification and implementation part

Sondre:

- Include more figures into the implementation, especially the calculation part.
- Finish writing about the Buck boost converter.
- Finish writing about the calculation regulator circuit.
- Write the design specification.

Eivind:

- Write about the configuration of the master
- Finish test reports
- Finish Ble protocol implementation

New meeting Monday 08.05.17

15th Meeting

Date: 08.May 2017
Who was at the meeting: Sondre, Jan Roar and Eivind.

From last time:
Group:

Sondre: I finished all the calculations and writing (tasks), have been working a lot on the report.

Eivind: The configuration of the master, finished test reports, finished ble protocol implementation, master and slave bubble diagram, master and slave, implementation.

Jan Roar: Missing just one function in the flowchart, but this will be finished today. Not finished theory part. Missing writing something about design patterns. UART test report is finished, also made a test on the TWI protocol on the slave.

Tasks until next time:

Jan Roar:

- Finish the last function in the flowchart, nus_data_handler
- Master implementations, write about the different threads.
- Master design specification, must write more here.
- Design Patterns, rewrite event driven and fsm. Write about RTOS and a combination of all three
- Go through validating and testing.
- Put the main.c, m_bus_receiver.c and .h in the report. And go through the comments.
- Fix the meeting report, summary and Validating & testing

Sondre:

- Include BOM into project.
- Project plan, hours and calculation.
- M-BUS read through after the removal of a step back.
- Finish writing on the Validation and testing.
- Fix the guidance meetings, and renewable meeting and meeting report writing.
- Clear faults in pictures, report.
- Write the discussion and Conclusion
- Validating and testing
- rewrite the theoretical part, switch mode power supply

Eivind:

- Problem statement and Problem solution
- Report outline
- Master implementations, write about the different threads
- convert the Man hour excel document to PDF
- Validating and testing
- Add lines between each row in the timesheets.

New meeting Thursday 11.05.17

16th Meeting

Date: 11.May 2017
Who was at the meeting: Sondre, Jan Roar and Eivind.

From last time:

Sondre: Created BOM, worked with project plan and fixed guidance meetings, wrote conclution and discussion.

Eivind: Problem statement, problem solution, timesheet. ++

Jan Roar: Finished the flowchart, but seems like i have to fix something. Finished design patterns.

Tasks until next time:

Jan Roar:

- Fix the uart part.
- Change the booble diagram for master, mind-map not booble
- Fix slave implementation code part, read done. And fix subsection
- Reference in master implementation software part. Clock reference?
- Maybe fix the master flowdiagram
- Go thru and fix reference and typos.

Sondre:

- Fix the work plan
- Fix the Hardware block diagram, should specify that PCB(Green) is one PCB
- Remove unnecessary figures on the hardware
- Testing and validating, more information about the testing and refer, not in a list.
- Add info about the AMS at the beginning of the discussion, why did we choose watt meter
- Mention AMS in the conclusion
- Change MBUS to M-BUS
- extend the abbreviation and sort it alphabetically
- Fix the appendix names
- Read through the report and fix bad writing and grammar
- write about master and slaves possibility to power the M-BUS

Eivind:

- 2.3 describe master and slave, Master provide slaved to the rest
- functional requirements at least one slave?
- 3.2.1 energy effective standard, is not a requirement?
- Test rapport
- Fix discussion
- Fix pressemelding med bilde
- Heap memory
- Litherature review

7.3.4 Gantt diagram

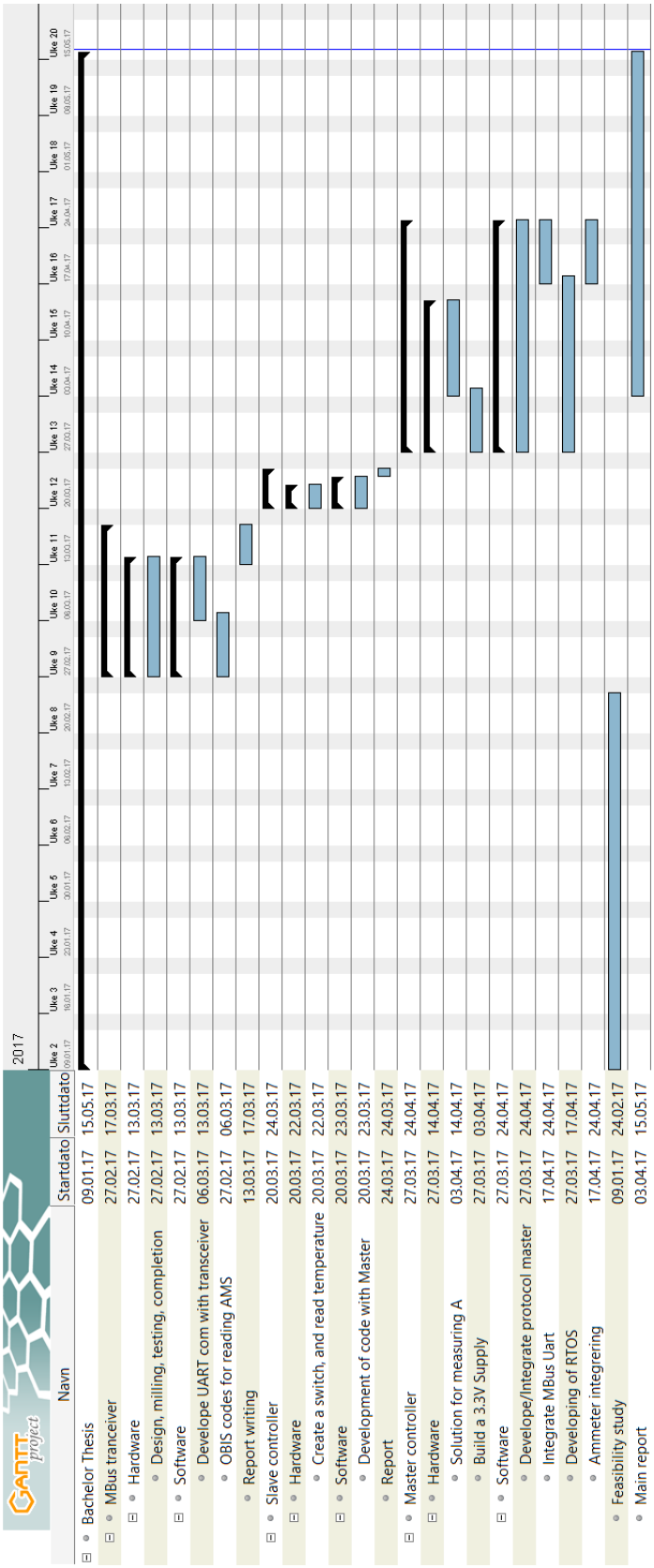


Figure 59: Gant diagram

7.3.5 Timesheet Eivind Stendal

Name/ week	Date	Hours	Description
Eivind Stendal	09.01.2017	5,00	Planning the project and start-up meeting
	10.01.2017	7,50	Planning the project and start-up meeting
	11.01.2017	5,00	Report- introduction
	12.01.2017	5,00	Fesability report
	13.01.2017	-	
	14.01.2017	-	
	15.01.2017	-	
Sum week 2		**22,50**	
	16.01.2017	11,00	Project description and planning
	17.01.2017	7,00	Meating with other group, project planning
	18.01.2017	-	
	19.01.2017	-	
	20.01.2017	-	
	21.01.2017	-	
	22.01.2017	-	
sum week 3		**18,00**	
	23.01.2017		
	24.01.2017		
	25.01.2017		
	26.01.2017		
			Meeting with renewable, project overview, and
	27.01.2017	5,00	meeting other groop, protocol, geir. Intro keil
	28.01.2017		
	29.01.2017		
sum week 4		**5,00**	
	30.01.2017	10,00	GCC compiler problems
	31.01.2017	8,00	BT protocol, GCC protocol
	01.02.2017	10,00	Rtos and SDK nrf52 research
	02.02.2017	10,00	Working with C code and SDL diagram for slave
	03.02.2017		
	04.02.2017		
	05.02.2017		
sum week 5		**38,00**	
	06.02.2017	7,00	Protocol and SLD diagrams
	07.02.2017		
	08.02.2017		
	09.02.2017	3,00	Meeting and worked with SLD diagram
	10.02.2017	6,00	Discussing algorith in master
	11.02.2017		
	12.02.2017		
sum week 6		**16,00**	

	13.02.2017	8,00	Flowdiagram of master and protocolresearch
	14.02.2017	5,00	Writing about protocol
	15.02.2017	9,00	Meeting, timesheet and protocol research
	16.02.2017	5,00	Reseach on obis and protocols
	17.02.2017	9,00	Research protocolls, and report finish
	18.02.2017	11,00	Finishing report +6h SDK reseach
	19.02.2017	-	
Sum week 7		**47,00**	
	20.02.2017	11,00	Finishing report and Nordic SDK ble
	21.02.2017	8,00	SDK research
	22.02.2017	8,00	ble sdk
	23.02.2017	2,00	ble_sdk
	24.02.2017	8,00	ble_sdk
	25.02.2017	-	
	26.02.2017	-	
Sum week 8		**37,00**	
	27.02.2017	8,00	ble sdk
	28.02.2017	10,00	ble sdk connection
	01.03.2017	11,00	ble sdk connection, ppi
	02.03.2017	6,00	Counter for slave w/ temp sensor
	03.03.2017	3,00	Git/GITHUB
	04.03.2017	-	
	05.03.2017	-	
sum week 9		**38,00**	
	06.03.2017	4,00	Git/ github problems++
	07.03.2017	8,00	ble, whitelist and peers
	08.03.2017	5,00	ble, whitelist and peers
	09.03.2017	5,00	ble, whitlist and peers
	10.03.2017	-	
	11.03.2017	-	
	12.03.2017	-	
sum week 10		**22,00**	
	13.03.2017	4,00	ble multipairing
			multipairing, giving up for now, central ->slave
	14.03.2017	7,00	comm
	15.03.2017	10,00	ble multiparing and protocoll
	16.03.2017	12,00	ble multiparing and protocoll
			Started on implementation of temp sensor in
	17.03.2017	6,00	slave code
	18.03.2017	-	
	19.03.2017	-	sick
sum week 11		**39,00**	

	20.03.2017	-	sick
	21.03.2017	-	sick
	22.03.2017	-	sick
	23.03.2017	9,00	finished temp sensor code, and tried to implement "central as slave"
	24.03.2017	5,00	implemented clock funtion in master, started to look at priority logarithm
	25.03.2017	7,00	ble protocoll and started to think about master logarithm
	26.03.2017	4,00	updated ble protocoll with ack
sum week 12		**25,00**	
	27.03.2017	10,00	Report -slave and protocoll. Master controller-code
	28.03.2017	-	
	29.03.2017	9,50	master protocol/ report
	30.03.2017	8,00	Report/ master, ble protocol phone ->master
	31.03.2017	7,00	app -master protocol is working
	01.04.2017	-	
	02.04.2017	-	
sum week 13		**34,50**	
	03.04.2017	12,00	Whitelist on central, some small adjustment on slave code and report work
	04.04.2017	-	
	05.04.2017	11,00	Merged the ble protocol code with controller and uart code. Also worked on flowdiagrams
	06.04.2017	-	
	07.04.2017	9,00	customize ble protocol for RTOS
	08.04.2017	8,00	customize ble protocol for RTOS
	09.04.2017	8,00	customize ble protocol for RTOS
sum week 14		**48,00**	
	10.04.2017	8,00	RTOS
	11.04.2017	2,00	RTOS problems
	12.04.2017	9,00	RTOS problems fixed, now working
	13.04.2017	-	Easter
	14.04.2017	-	Easter
	15.04.2017	-	Easter
	16.04.2017	-	Easter
sum week 15		**19,00**	

	17.04.2017	-	Easter
	18.04.2017	-	
	19.04.2017	9,00	Merged new UART code with controller, changed referances to IEEE
	20.04.2017	8,00	report, references and ble protocol
	21.04.2017	5,00	Added funtionality to the nos phone -master protocoll
	22.04.2017	-	
	23.04.2017	-	
sum week 16		**22,00**	
	24.04.2017	11,50	Report, about ble protocol, phone ->master prot
	25.04.2017	7,00	Finished phone-> master protocol, started to work on test checklist and cleaned up code.
	26.04.2017	5,00	Testing software
	27.04.2017	6,00	Testing software
	28.04.2017	10,00	Testing software
	29.04.2017	10,00	Testing software
	30.04.2017	7,00	Testing software
sum week 17		**56,50**	
	01.05.2017	7,00	Preface, ble protocol
	02.05.2017	7,00	Report ble, diagrams
	03.05.2017	9,00	Report, meeting and fixing on the report based on the feedback we got on meeting.
	04.05.2017	8,00	Master and slave bubble diagram. Master and slave implementration diagram
	05.05.2017	7,00	Report
	06.05.2017	-	
	07.05.2017	-	
sum week 18		**38,00**	
	08.05.2017	11,00	Fixing on introduction
	09.05.2017	10,50	Press release, introduction, fix timesheet.
	10.05.2017	-	Prepearing for exam proparbility
	11.05.2017	6,00	Finish report
	12.05.2017	10,00	Finish report
	13.05.2017	10,00	Finish report
	14.05.2017	10,00	Finish report
sum week 19		**57,50**	

7.3.6 Timesheet Sondre Håverstad

Name/week	Date	Hours	Description
Sondre Håverstad	09.01.2017	5,00	Planning the project and start-up meeting
	10.01.2017	7,50	Planning the project and start-up meeting
	11.01.2017	5,00	Feasibility study report - issue, Altium Designer study.
	12.01.2017	6,00	Feasibility report and research M-Bus
	13.01.2017	-	
	14.01.2017	-	
	15.01.2017	-	
sum week 2		**23,50**	
	16.01.2017	6,00	Project description and planning
	17.01.2017	7,00	Meating with other group, project planning
	18.01.2017	-	
	19.01.2017	-	
	20.01.2017	-	
	21.01.2017	-	
	22.01.2017	-	
sum week 3		**13,00**	
	23.01.2017	-	
	24.01.2017	-	
	25.01.2017	-	
	26.01.2017	-	
	27.01.2017	5,00	Meeting with renewable, project overview, and meeting with the other groop, protocol, Mbus, Obis-codes, current sensor
	28.01.2017		
	29.01.2017		
sum week 4		**5,00**	
	30.01.2017	6,00	Working on ordering components needed for measuring ampere and m-bus Tranceiver. Also done some research on both fields.
	31.01.2017	-	
	01.02.2017	10,00	Worked to find components and parts for the project, discussed solutions and started the design of MBUS transceiver
	02.02.2017	-	
	03.02.2017	-	
	04.02.2017	9,00	Wrote about AMS and MBUS receiver in the feasibility study. Didresearch and designed first version of MBUS receiver in Altium.
	05.02.2017	5,00	Wrote on issue and execution plan part of the, but did noet finish the issue
sum week 5		**30,00**	

	06.02.2017	5,00	Wrote the rest of Issue on feasibility study and worked on M-Bus receiver design.
	07.02.2017	2,00	Research on M-bus and wrote a theoretical background on M-bus.
	08.02.2017	8,00	Did some research on M-Bus and found documents ablut the theme. Did work some one the design of the tranceiver.
	09.02.2017	3,00	Meeting and Worked on Gant diagram.
	10.02.2017	6,00	Discussion about the master unit, and did some work on the Gantt diagramm and SWOT analysis
	11.02.2017		
	12.02.2017		
sum week 6		**24,00**	
	13.02.2017	8,00	Helped with the Flowdiagram and did some research on the ammeter
	14.02.2017		
	15.02.2017	6,00	Worked on the Gantt diagram, and did some research.
	16.02.2017	3,00	Worked on the workplan and wrote about extensions + research.
	17.02.2017	9,00	worked on extensions and master, and correcting the feasibility report.
	18.02.2017	5,00	Completion of feasibility study report, correrccting, rewriting and such.
	19.02.2017		
sum week 7		**31,00**	
	20.02.2017	9,00	Finished Feasibility study and started with the MBUS trans design.
	21.02.2017	-	
	22.02.2017	6,00	Group meeting and MBUS design + protocol
	23.02.2017	-	
	24.02.2017	7,50	Worked on the Mbus design, and made it ready for the milling machine.
	25.02.2017	6,50	Did some small changes on the design and wrote some on the report. + research on mBus
	26.02.2017	-	
sum week 8		**29,00**	
	27.02.2017	3,00	Some small changes on the design and i did send it to print. Home from school, did som work.
	28.02.2017	-	
	01.03.2017	8,00	Got the board and started soldering, + research on the slave device.
	02.03.2017	6,00	Did the last soldering on Mbus tranceiver, and started a design on the slave device.
	03.03.2017	-	
	04.03.2017	-	
	05.03.2017	-	
sum week 9		**17,00**	

	06.03.2017	5,00	Finished design of the slace device, and sent it
	07.03.2017	-	
	08.03.2017	5,00	Research on the new watt meter, and research slave.
	09.03.2017	1,00	Research relay slave.
	10.03.2017	8,00	Slave research and ordering some components for master and slave. Worked on the design slave.
	11.03.2017	-	
	12.03.2017	4,00	Did some work on the Slave and looked for components to use.
sum week 10		**23,00**	
	13.03.2017	3,00	Worked on a solution for the slave, most research.
	14.03.2017	1,00	Research slave.
	15.03.2017	10,00	Did some research on the Mbus wattmeter + worked on a solution to powering the mbus. Ordered parts.
	16.03.2017	-	
	17.03.2017	6,00	Meeting, discussed mBus powering, tested the Mus tranceiver to see if it worked, debug later. I found another solution to powering that may worked, i built it but not tested.
	18.03.2017	2,00	research mbus and preparation before testing mbus powering.
	19.03.2017	5,00	Worked on the simple MBUS traceiver for testing the Watt-meter with MBUS interface. Simulated an buildt the design.
sum week 11		**27,00**	
	20.03.2017	8,00	Tested the Simple MBUS design with Jan Roar, we got it to work and did manage to receive the information from it. I started creating components for PCB design in Altium. Ready to create it.
	21.03.2017	2,00	Finished drawing the Simple MBUs tranceiver, it is now ready for print. There only a few practical things to check tomorrow.
	22.03.2017	7,00	Sendt the simple mbus to printing and tested it once more. Got the parts for the slave device, imported models to altium. And worked on design.
	23.03.2017	-	Sick
	24.03.2017	-	Sick
	25.03.2017	-	Sick
	26.03.2017	-	Sick
sum week 12		**17,00**	

	27.03.2017	8,00	Tried to solder the finsihed simple mbus traceiver, but it failed, i have done some mistakes that i should have known about, so i was forced to createate a new one. That i did, and sendt it to print.
	28.03.2017	3,00	Created some block diagrams over the Hardware and wrote some on the MBUS under Theoretical background.
	29.03.2017	8,00	I got the new pcb, i did solder two simple mbus tranceivers and both worked fine. I made a adapter for USB to usb for the Master and a adapter that let ut mount the nrf52 on DIN-rail. I also made the 12V-to 3.3V regulator and sendt it to print.
	30.03.2017	2,00	Research Uart and looking for a solution for permanently powering MBUS with 36V.
	31.03.2017	6,00	Soldered the USB adapter, the 12V to 3.3V converter and got it working. And buildt the master / wattmeter box with the component avaible
	01.04.2017	-	
	02.04.2017	4,50	Think i found a solution to the mbus powering problem, and worked on Block diagram over the hardware in lucidchart.
sum week 13		**31,50**	

	03.04.2017	10,00	Tested the mbus tranceiver for fault, found out the fault was in the code. Tested the relay with swich against the nrf52. integrated all the circuits into a design and sendt it to print. Worked with the Bachelor report, made some pictures.
	04.04.2017	-	
	05.04.2017	6,00	Worked on the report, i wrote about the simple mbus tranceivier and workedon the Master design.
	06.04.2017	4,00	Worked on the report and wrote about the master and did some work on the Master design.
	07.04.2017	6,00	Worked on th report, re-wrote the test report and about the master hardwae solution. Did some research on buck boost.
	08.04.2017	11,00	Isoldered two slaved today, 12V to 3.3V regulator worked, 12V to 9V regulator also worked without circuitry faults. But teh 3v3 powering of nRF52 failed, i think i killed a nrf... did some research and worked ona solution.
	09.04.2017	-	
sum week 14		**37,00**	
	10.04.2017	2,00	Worked on designing the master
	11.04.2017	3,00	Worked on designing buck boost converter master and wrote a test report from notes into the report.
	12.04.2017	7,50	Meeting with the group and planning. I fixed some faults on the slave design but we did not manage to get the temperature sensor working.
	13.04.2017	3,00	Wrote some on the lab reports and worked on some faults in the Altium master design.
	14.04.2017	5,50	Worekd on the report and developing of master design.
	15.04.2017	2,00	Worked on the report.
	16.04.2017	5,00	Worked on the report, bust most of the time worked on Buck boost design, master.
sum week 15		**28,00**	

	17.04.2017	5,00	Worked on the report and , almost finished the Master design. It may be ready for print this week.
	18.04.2017	-	
	19.04.2017	10,00	Completed the master design, and sendt it to print,i found out the buck boost converter with help form supervisors. Found a fault on slaves, one bad solder and one defect component. We also did a meeting with the supervisors.
	20.04.2017	-	
	21.04.2017	4,00	Worked on the report, Master design.
	22.04.2017	5,00	Worked on the report, Schematic and slave.
	23.04.2017	3,00	Worked on the report, Lab test report and i did rewrite some on MBUS.
sum week 16		**27,00**	
	24.04.2017	8,00	Soldered the master, it is now ready for testing, we had a group meeting.
	25.04.2017	-	
	26.04.2017	9,00	Tested the master design with Eivind, the rest of the test will be tested on Friday. Integrated master into fuse box with DIN rail mount. Wrote on the slave part on the report.
	27.04.2017	2,00	Worked on the repost, slave hardware.
	28.04.2017	6,00	Tested the master and meeting about the report, we made some modifications to the outline. Wrote some on the slave hardware.
	29.04.2017	2,00	Relay slave design in the report.
	30.04.2017	5,00	Worked on the report, master design.
sum week 17		**32,00**	

	01.05.2017	6,00	Worked on the report, problem definition, background and design spesefication
	02.05.2017	-	
	03.05.2017	9,00	Meeting with supervisors and writing on the report.
	04.05.2017	2,00	Worked on the report
	05.05.2017	6,00	Worked on the report
	06.05.2017	-	
	07.05.2017	4,00	Worked on the report
sum week 18		**27,00**	
	08.05.2017	10,50	Worked on the report, and group meeting. Started on the discussion and finished the meeting reports.
	09.05.2017	8,00	Worked on the report, discussion, calculations.
	10.05.2017	8,00	Worked on the report, finished for deliverance to the supervisors
	11.05.2017	11,00	Worked on the report and meeting with supervisors.
	12.05.2017	10,00	Finish report
	13.05.2017	10,00	Finish report
	14.05.2017	10,00	Finish report
sum week 19		**67,50**	

7.3.7 Timesheet Jan Roar T. Mydland

Name	Date	Hours	Description
Jan Roar Mydland	09.01.2017	5,00	Planning the project and start-up meeting
	10.01.2017	7,50	Starting to check out free Rtos and project panning
	11.01.2017	2,00	Studying FreeRTOS, by reading and watching clips on youtube
	12.01.2017	2,00	Studying FreeRTOS, by reading and watching clips on youtube
	13.01.2017		
	14.01.2017		
	15.01.2017		
sum week 2		**16,50**	
	16.01.2017	6,00	Project description and planning
	17.01.2017	7,00	Meating with other group, project planning
	18.01.2017	-	
	19.01.2017	-	
	20.01.2017	2,00	Some studying of nRF52, and the nRF52832 chip
	21.01.2017	-	
	22.01.2017	-	
sum week 3		**15,00**	
	23.01.2017	-	
	24.01.2017	-	
	25.01.2017	-	
	26.01.2017	-	
	27.01.2017	4,00	Meeting with renewable, project overview, and meeting with the other group, protocol, Mbus, Obis-codes, current sensor. And some studying of the nrf 52 datasheet.
	28.01.2017		
	29.01.2017		
sum week 4		**4,00**	
	30.01.2017	6,00	Studying of BLE. And starting to look on GCC in order to compiles larger than 32kB.
	31.01.2017	8,00	
	01.02.2017	12,00	Reading about RTOS, and finding out how to use it.
	02.02.2017	-	
	03.02.2017	-	
	04.02.2017	-	
	05.02.2017	-	
sum week 5		**26,00**	

	06.02.2017		
	07.02.2017		
	08.02.2017	4,00	Some more reading about FreeRTOS.
	09.02.2017	6,00	Finished reading about FreeRTOS. Started writing on the feasibility study report.
	10.02.2017	5,00	Started thinking about the SDL diagram. And discussion in group about what to do.
	11.02.2017		
	12.02.2017		
sum week 6		**15,00**	
	13.02.2017	8,00	Working on the float chart of the master central. And some resource about timer,uart,saadc on the nrf52 documentation.
	14.02.2017	8,00	Writing in the report about FSM.
	15.02.2017	8,00	Meeting in the beginning of the day. Finishing the flowdiagram of both central device and slace device. Some more writing in the feasibility study report.
	16.02.2017	1,00	Some resource about nrf52
	17.02.2017	-	
	18.02.2017	7,00	Writing in the report about algorithm.
	19.02.2017	3,50	Finish writing about FSM and RTOS. Think the feasibility study report is almost finish now.
sum week 7		**35,50**	
	20.02.2017	-	
	21.02.2017	-	
	22.02.2017	8,00	Meeting with the group. And starting making a uart dummy from an arduino
	23.02.2017	6,00	Continued with the uart dummy. And reading about the obis protocol
	24.02.2017	8,00	Manage to get connection over uart, and send it to my phone via ble.
	25.02.2017	-	
	26.02.2017	-	
sum week 8		**22,00**	
	27.02.2017	1,00	Some small reasource about freertos
	28.02.2017	-	
	01.03.2017	11,00	Implemented FreeRTOS with a master BLE controller.
	02.03.2017	-	
	03.03.2017	-	
	04.03.2017	-	
	05.03.2017	1,00	Started creating task for uart event handler.
sum week 9		**13,00**	

			Some github work, and some more studying of BLE to figure out what is realy happening. And to make sure we can get FreeRTOS to work
	06.03.2017	9,00	fine.
			First some more studying about BLE. Then some work on the FreeRTOS, activatet nrf_logg. And then I got a lot of errors. Trying
	07.03.2017	9,00	to figure out what it is.
			Some more figuring out what is happening
	08.03.2017	5,00	with the FreeRTOS in the central device.
			Found out that it wasn't softdevice that caused the problem. But the implementation of the uart module. Found how to make larger heap
	09.03.2017	9,00	size in order to get more task
			Found out the problem. Had to init uart before logging. Picked up the m-bus reader from school. And started planning the uart implementation. Made a .c and .h file for the
	10.03.2017	7,00	uart implementation.
			Some small writing in the kladd, and startet
	11.03.2017	2,00	planning the m-bus implementation
			Some small writing in the kladd, and startet
	12.03.2017	2,00	planning the m-bus implementation
sum week 10		**43,00**	
			Some more implementation of the m_bus
	13.03.2017	2,00	receiver code
			Some more implementation of the m_bus
	14.03.2017	2,00	receiver code
			Implementation of m_bus receiver. Setting
	15.03.2017	12,00	opp a logic logger. Meeating with supervisors.
			Continued with the m_bus_receiver implemantation. Metting with the other groups at renewable. Made a dummy
	16.03.2017	13,00	m_bus_receiver out of another nrf52
			Meeting in the group, discussed solution on the m_bus power with Sondre. Some more work on the m_bus uart module. Studying uart app module on nrf52. And figuring out mutex
	17.03.2017	6,00	and semaphores in the freertos.
	18.03.2017	6,00	Continued on the m_bus uart receiver .
			Continued on the m_bus uart receiver . Trying to figure out how to receive uart rx events.
	19.03.2017	2,00	Thinking about using a queue.
sum week 11		**43,00**	

			Working on the receiving of m_bus items over uart. Think I have figured out a neet design. Starting to see if it works. Also testet the
	20.03.2017	9,00	m_bus meter with Sondre. Just som uart work, trying to figure out why
	21.03.2017	3,00	the m_bus dummy nrf52 wont work. Testint central ageinst our m_bus receiver. Did order to get response, and save it. And then send it to the controller task. Startet working
	22.03.2017	6,50	on the flow chart, and some notes.
	23.03.2017	1,00	
	24.03.2017	-	
	25.03.2017	7,00	Working on the algoritm of the central device. Fixing the box with the m_bus receiver. Implementet some new functions in the m_bus receiver file. Fixed the header file for
	26.03.2017	3,00	m_bus_receiver.
sum week 12		**29,50**	
	27.03.2017	8,00	Continued working on the uart.
	28.03.2017		
	29.03.2017	6,00	Working on the uart, meeting in the group. Testing m_bus
	30.03.2017	-	
	31.03.2017	7,00	Working on the uart, and testing against the m bus meter. Seems like it worked fine Had major problems with the m bus, did get error codes on the nrf52. Trying to figure out what was wrong. It is the uart callback that
	01.04.2017	5,00	triggers the message. Trying to figure out what it is that triggers the fault. My conclusion is that the RX pin on nrf52 is always logic '0'. And it should be logic high or
	02.04.2017	5,00	'1'. And some small work in the report.
sum week 13		**31,00**	

	03.04.2017	-	
	04.04.2017	1,00	Just looking on the code to see if something looked ud in the uart. Didn`t seem like that troubleshooting the uart to m_bus converter. Found out that one of the input pin wasn't connected to anything. Fixed a short cut, then everything worked. The uart worked. Merged my file with Eivinds file. Looked like we are getting pretty closed to a first test demo
	05.04.2017	11,00	project.
	06.04.2017	8,00	Working on flow chart of the uart module
	07.04.2017	8,00	Testing and discussing freertos with Eivind. Figured out a new way of implenting the uart. Started the new way of implementing the uart.
	08.04.2017	6,00	Was reading in the m bus lecture.
	09.04.2017	-	
sum week 14		**34,00**	
	10.04.2017	8,00	working on uart.
	11.04.2017	-	
	12.04.2017	6,00	meeting in the group. Continued with uart.
	13.04.2017	1,00	Flowchart uart
	14.04.2017	1,00	Flowchart uart
	15.04.2017	1,00	Flowchart uart
	16.04.2017	-	
sum week 15		**17,00**	
	17.04.2017	-	
	18.04.2017	11,00	Flowchart uart, and writing in the theory part of the report.
	19.04.2017	11,00	Coding, and testing. Including meeting in the group. And with the supervisors.
	20.04.2017	2,00	Work on flowchart
	21.04.2017	2,00	Work on flowchart
	22.04.2017	2,00	Work on flowchart
	23.04.2017	4,00	Work on flowchart, and report writing
sum week 16		**32,00**	
	24.04.2017	8,00	Working on the flowchart. Meeting in the group.
	25.04.2017	-	
	26.04.2017	-	
	27.04.2017	7,00	Work on the flowchart
	28.04.2017	8,00	Work on the flowchart, got thru the report with the group
	29.04.2017	7,00	Work on the flowchart. Finish controller_task
	30.04.2017	-	
sum week 17		**30,00**	

	01.05.2017	6,00	Worked on the report. Theory part.
	02.05.2017	8,00	Worked in the report, theory part.
	03.05.2017	8,00	Worked in the report, theory part. And meeting in the group and with the supervisors.
	04.05.2017	11,00	Go thru the code with Eivind. And discussed some flowchart etc.
	05.05.2017	7,00	Meeting in the group. Made some test report, regarding uart and twi with the logic analyzator.
	06.05.2017	2,00	Flowchart.
	07.05.2017	8,00	More flowchart. Almost fineshed it.
sum week 18		**50,00**	
	08.05.2017	4,00	Finished the flowchart. Meeting with the group.
	09.05.2017	10,00	Writing in the report, and found out that not all the flowchart wasn't finished. So started drawing the last one.
	10.05.2017	-	
	11.05.2017	9,00	Finish report
	12.05.2017	10,00	Finish report
	13.05.2017	10,00	Finish report
	14.05.2017	10,00	Finish report
sum week 19		**53,00**	

7.3.8 Timesheet Total

	Total hours	%
Eivind	583,00	36,85 %
Sondre	489,50	30,94 %
Jan	509,50	32,21 %
Tot	1 582,00	
Weeks since start	17,71	
Avg hours/weeks	89,31	
Avg hours/weeks/person	29,77	

7.4 Appendix C - Press release in Norwegian

Pressemelding

Contact: Eivind Stendal
Lian Platå 31 4638
Kristiansand, Norge
Phone: (+47) 991 66 323
Email: eivindstendal@gmail.com
Date: 13. mai 2017

UIA studenter med smart system til AMS måleren

Når strømleverandørene innfører overpris og høyere priser på morgen og ettermiddag kan dette systemet være gunstig for deg.

Grimstad, Norge – Juni 1,2017 –Fremføring av bachelor prosjekt på UIA

Strømleverandørene ønsker å jevne ut strømforbruket i Norge, og nå som smarte strømmålere (AMS) er innført i de fleste norske hjem, har de muligheten til å presse kundene til å tenke på dette ved bruk av høyere pris ved overforbuk og høyere priser når det gjennomsnittlige forbruket ligger høyest. Det er typisk på morgenen og ettermiddagen at forbruket er høyest.

Vår oppgave løste dette problemet vet å skru av apparater som det ikke er så viktig at står på i disse dyre tidspunktene/ tilstandene uten at det skal gå ut over komforten til kunden. Et eksempel kan være at varmtvannsberederen blir slått av mens du støvsuger for ikke å betale for overforbruk, men den vil kun bli slått av om temperaturen i berederen er tilstrekkelig høy. Et annet eksempel er at en ovn blir slått av på dagen når strømmen er dyrest hvis det er mindre enn 2 grader avvik fra ønsket temperatur i rommet.

Apparater som typisk styres er varme-apparater som varmtvannsbereder eller panelovner. For å sette opp systemet trenger man ikke fylle huset med nye kabler, alt av kommunikasjon mellom sentralenheten og apparatene vil gå trådløst.

Deltagere på prosjektet:

Eivind Stendal - Software
Jan Roar T. Mydland - Software
Sondre Håverstad - Hardware

# # #

7.5 Appendix D - Test reports

7.5.1 Testing of the simple M-BUS transceiver circuit

Sondre Håverstad and Jan Roar Mydland
Date: 20.03.17

Introduction/Purpose Since were not able to get the AMS smart meter in time, we was forced to look for alternatives to show the principle of out system and get the project ready in time. Since we don't get the AMS smart meter in time, we ordered a wattmeter with M-BUS interface to show the principle of our system.

The First M-BUS tranceiver that was made is more complex and is not tested since we don't have the AMS. the M-BUS tranceiver also need external powering from the AMS to work. The new wattmeter is also a slave that needs external powering, and the solution will be to build a simple master that can both tranceive and receive from UART to M-BUS, plus power the bus. And have some of the key function as the AMS smart meter does.

The new wattmeter doesn't send data before it gets a request from the μC, and it do not respond to unknown queries like for instance wrong address.

Materials

- A circuit based on the one from Github.
- Power supply with 32-36VDC output.
- A μC that communicates through UART (Using the nRF52 μC).
- Salea logic analyzer with related computer software.

Methods

- We started to test the design by simulating it, and it seems to work, design shown in figure 60.
- We used the oscilloscope to check the TX and RX pin, and it seems like the Tranceiver is sending something but with a length of 2-4uS, which is to short.. shown in figure 61a.
- We measured around the circuit and i think the last transmitting transistor is the problem.
- We measure that the last transistor don't really put out the signal it receives on the base. We tried to set Down the baud-rate to 2400 baud from 9600.
- Found out it was a miss connection on the transceiver, PNP transistor, switched Base and emitter. It seems to work fine and as intended now.
- We connected the watt-meter to the circuit and configured the right setup and right address of the watt-meter, it responded nicely. Response shown in figure 61b and 62a. Figure 66 shows sending init on all available addresses, and after that it shows the request from master and response from wattmeter. Figure 65 shows the response from wattmeter after initialising on the correct address.
- This simple transceiver causes some echo on the RX pin when transceiving, shown in figure 62b, but the MicroController can only transmit or receive at the time. Figure 63 shows the echo on the TX pin when RX pin on the microcontroller are sending. On figure 64 we can see that the length of the echo is only about $6 - 7\mu S$, and it seems like this is not a problem.
- The last figure 67 shows that the new wattmeter don‘t respond to unknown queries.

Conclusion This test shows that the simple M-BUS transceiver works and can be used in out circuit. It is ready to be integrated into the master design. This test also shows that sending request on the correct address gives us all the data from the new wattmeter. This also shows that sending init (or request) on the wrong address don‘t give a response.

References I (Sondre) found this example online at Github:
`https://github.com/rscada/libmbus/blob/master/hardware/MBus_USB.pdf`

More information on the Salea logic analyzer can be found here:
https://www.saleae.com/

Figures & Graphs Simulation circuit and results.

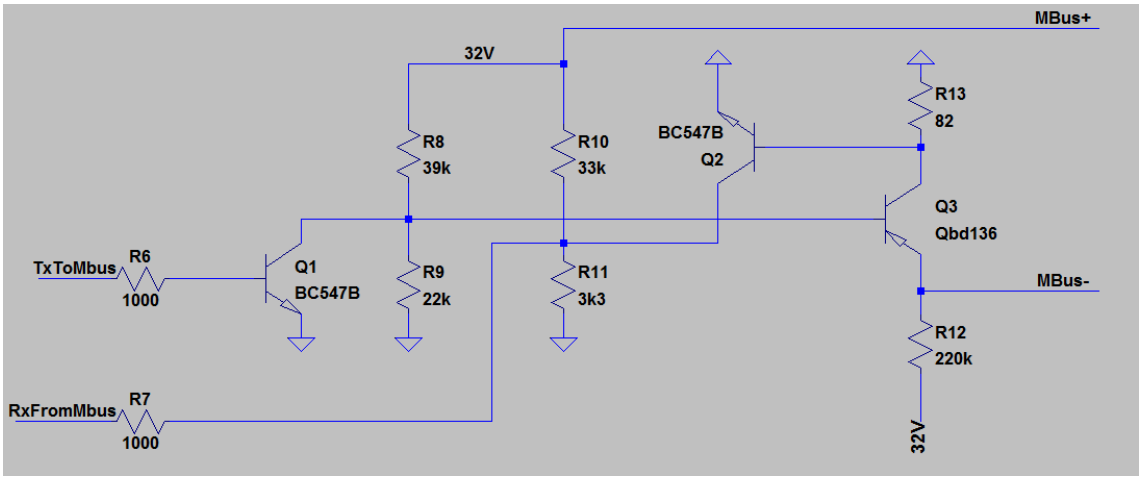
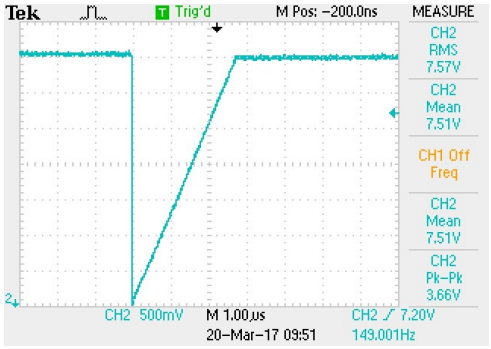
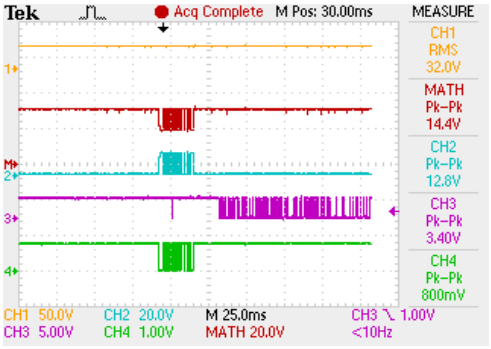


Figure 60: Simple M-BUS tranceiver simultaion

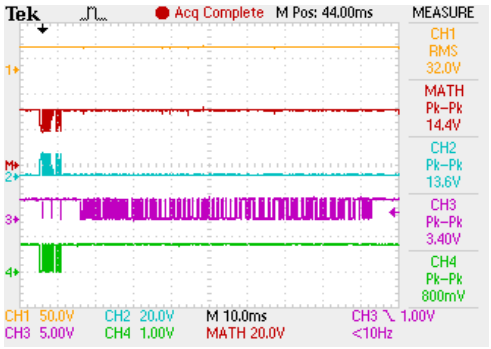


(a) Fault sending, simple M-BUS traniceiver

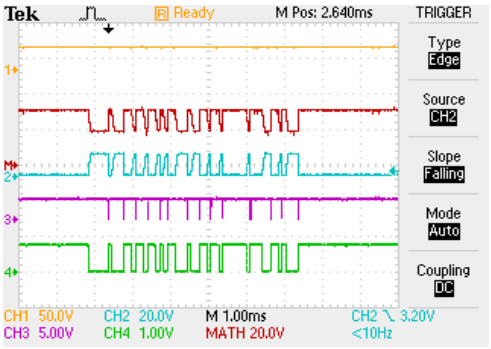


(b) Watt-meter respense(Purple) 1

Figure 61: Measurement M-BUS tranceiver



(a) Watt-meter respense(Purple) 2



(b) Echo on RX pin when Trenceiving

Figure 62: Measurement M-BUS tranceiver 2

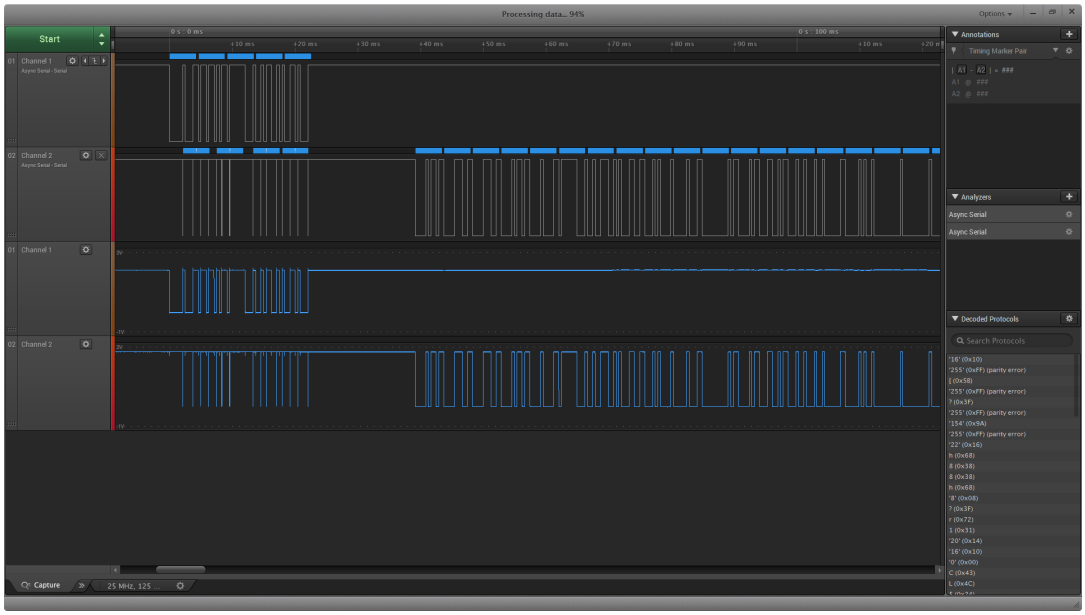


Figure 63: Echo seen on the logic analyzer

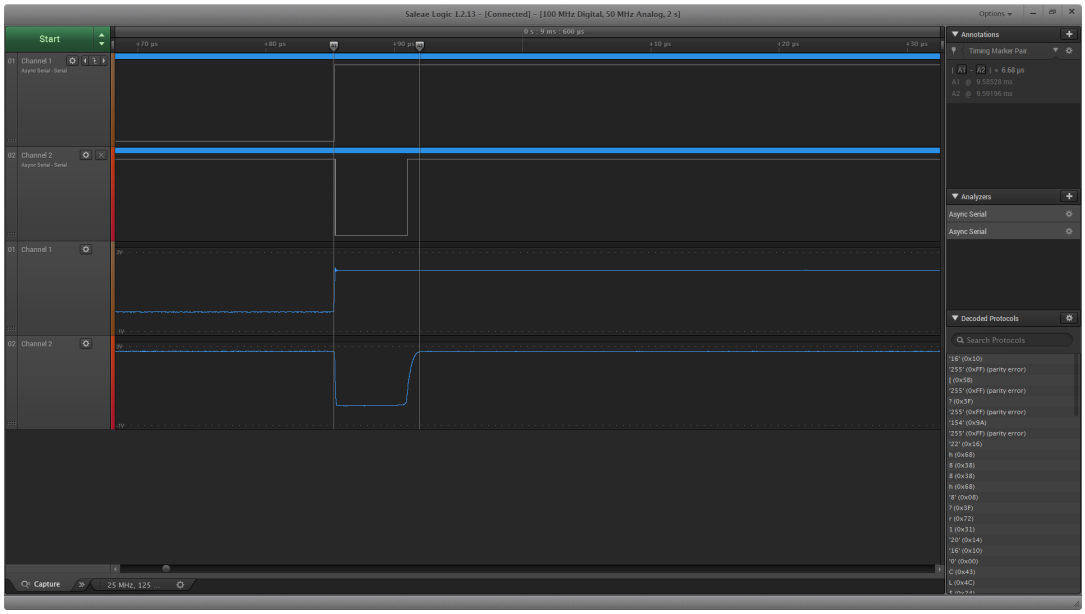


Figure 64: Length of the echo seen on the logic analyzer

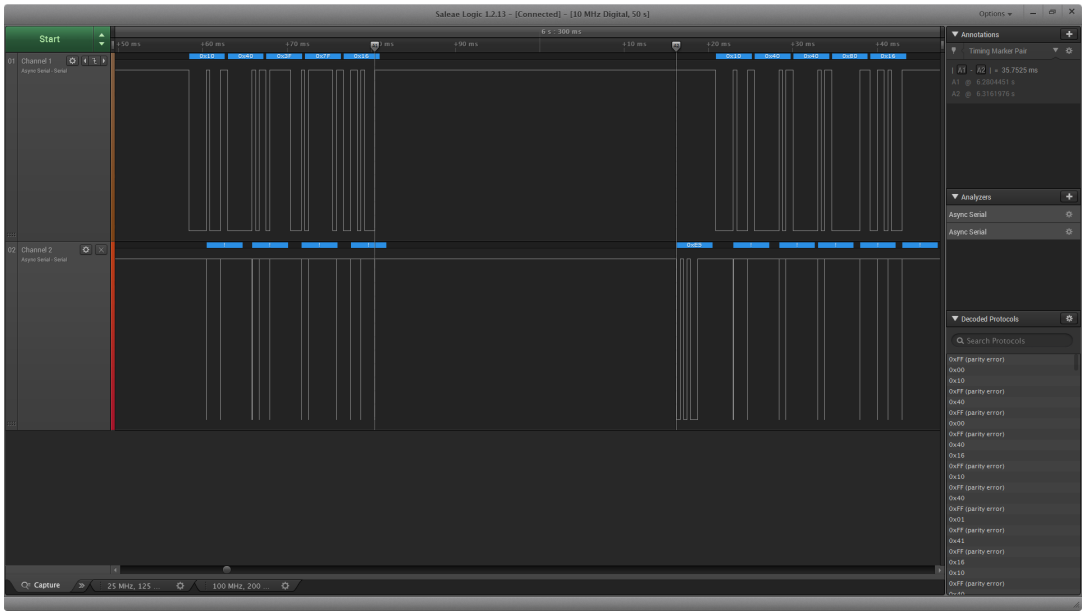


Figure 65: Sending init on correct address, and get response as seen on the logic analyzer

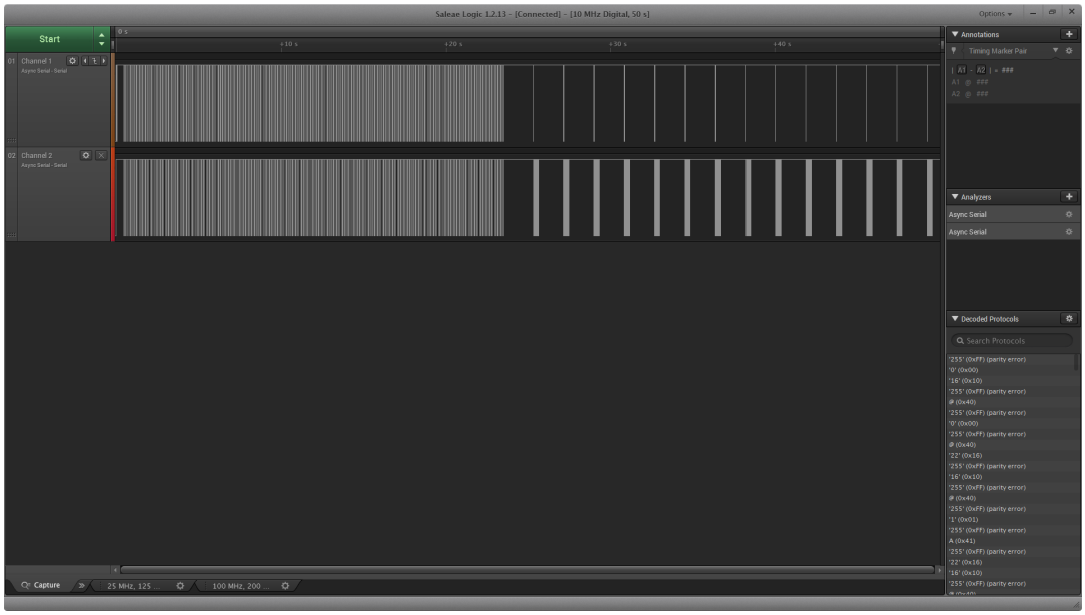


Figure 66: Sending init on all addresses, and then sending request as seen on the logic analyzer

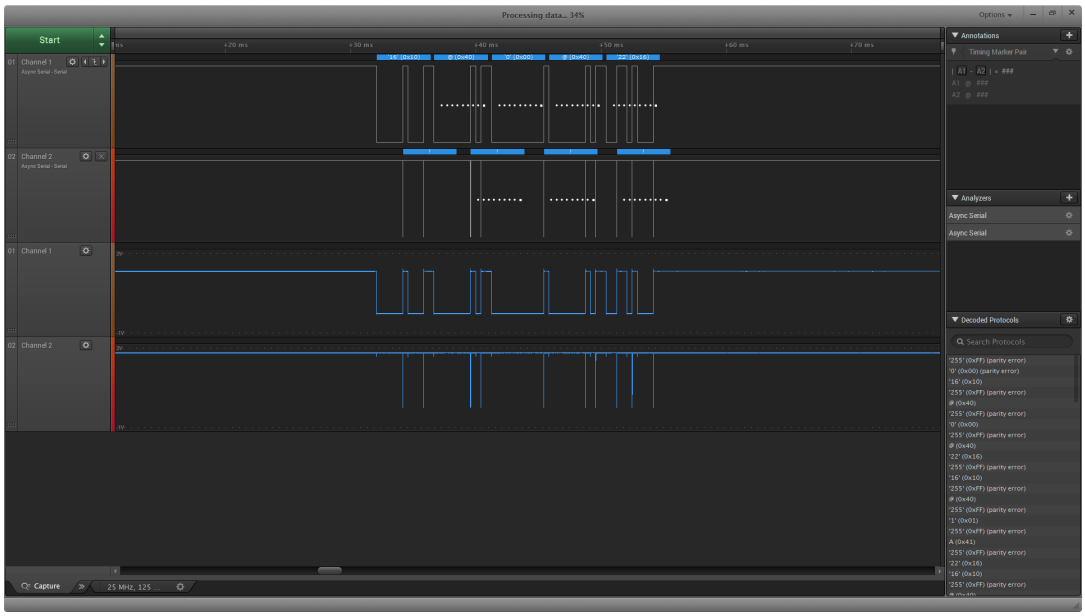


Figure 67: Sending init to a wrong address and no response as seen on the logic analyzer

7.5.2 Testing of the LM317 - 12V to 3v3 regulator circuit

Sondre Håverstad
Date: 30.03.17

Introduction/Purpose A 12V to 3V3 regulator would be beneficial for the project, in order to keep the micro controllers operative without USB or any other external power supplies. We Intend to use a 230VAC to 12DC transformer and regulate that down to 3v3 in order to power nrf52.

Materials

- Power supply with a 12Vdc output.
- A oscilloscope or multimeter to do measurements with.
- Test probes.
- A breadboard.
- LT1072
- Electrolyte capacitors: 100uF, 10uF.
- Ceramic capacitors: 100n.
- 3 diodes: 1N002.
- Resistors: 300, 470.

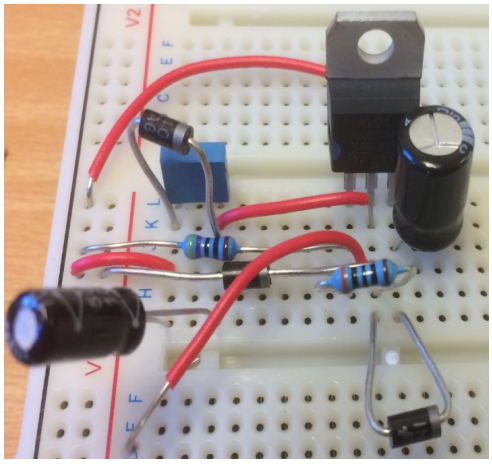
Methods

- Drew the circuit in LTSPice and got it working, shown in figure 70a, and the results at figure 71a.
- When the simulation works, construct the circuit shown in figure 70a on the breadboard. Breadboard circuitry shown in figure 68a.
- Measured the output it did not work, and found ut the capacitor C1 and C2 was connected to ADJ pin of LT1072 and not ground. Fixed it and it worked fine.
- Applies Any voltage between 12V and down to 5V as shown in figure 69a, and it kept a stable 3v3 output.
- At first resistor R2 equal to 520 Ω, creating a output at 3.6V, which is to close to what nRF52 can handle. so changed R2 to 470 to get Output voltage a little bit lower to be sure, down to 3-3v3.

Conclusion These measurements shows that the regulator circuit works fine and with the intended output. It should be ready for integration into both master and slave design.

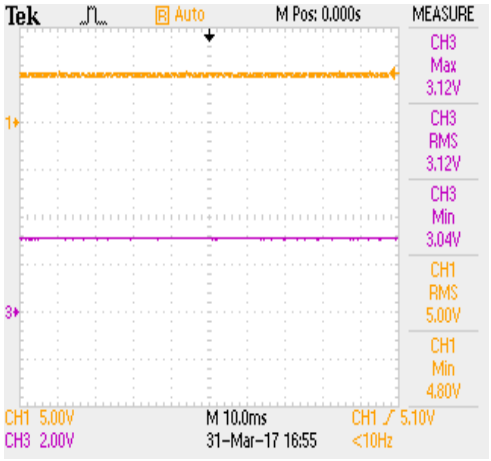
References The test circuit was taken from the application proposal in the data sheet of the LM317, Only the values got redesigned [42].

Figures & Graphs Simulations and results.



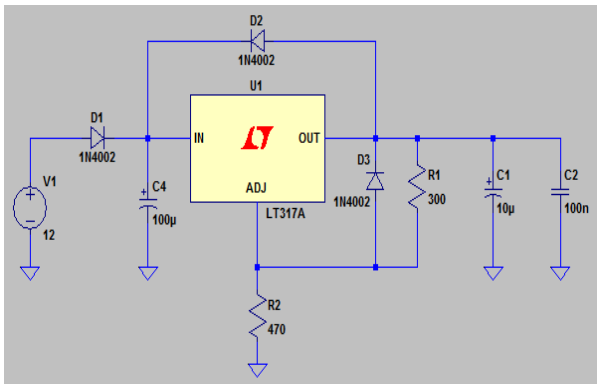
(a) Testing of the 3.3V regulator on breadboard

Figure 68: Measurement regulator 3V3



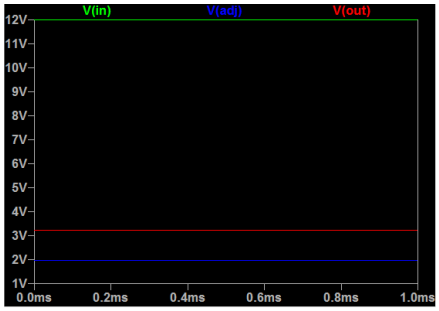
(a) Simulation results of the 3v3 regulator

Figure 69: Measurement regulator 3V3 2



(a) 12V to 3V3 regulator circuit

Figure 70: Measurement regulator



(a) Simulation results of the 3v3 regulator

Figure 71: Measurement regulator 2

7.5.3 Testing of the complete slave device circuit

Sondre Håverstad, Eivind Stendal
Date: 08.04.17

Introduction/Purpose The slave device circuit has a 12V input which comes from a external 230VAC to 12Vdc transformer. After the connector on the board it splits into two regulators, one 3v3 regulator to power the board and the other is a 9V regulator which will be used to trigger a relay though a transistor switch. There is also a LM75B temperature sensor on the board.

Materials

- Power supply with 12VDC output.
- A microController with I^2C and GPIO.

Methods

- After soldering, i connected 12Vdc to the board to check of i got the right values as output.
- Tune the power supply to 12Vdc and connect it to the board.
- Measured the the output of both regulators and got 3v3 and 9.6V, and it seems to be working.
- Make a connection between P0.11 to 3v3, this will trigger the relay to close .
- There is a fault with the relay because it wont trigger.
- Measuring over the coil of the relay and it does not see any voltage.
- I found the fault. I missed one 0Ωresistor between the regulator and the relay.
- I discovered that i have forgotten to add a reversed voltage protection to the +12V input. i mount a diode in this pin.
- Measured the voltage at the temperature sensor and it got 3v3 from the regulator.
- The PCB seems to be working as planned and is ready to be connected to the micro-Controller.
- Connect it to the controller and power it up.
- nRf52 wont power up... i think i killed it.
- 3v3 supply have been connected to VDD and i down know why this doesn't work. I have been looking at the hardware files for a while [7.5.3] and i think that i should connect the regulator to the external power pins at the edge instead of the 3v3 pin-out, because nRF52 has some internal protection then and a own regulator.
- Made some modifications to the PCB, i removed the direct connection from 3v3 regulator to the board and made two wires for connection to the external pins at the edge.
- power it up by again and the nrf52 now run without a battery or USB connected.
- Keil cashed after connection of the PCB.
- I checked all connection on the board and found out there must have been a connection from one of the connectors to ground that i wasn't able to see. For after i tried to check the spacing between with a screwdriver , the fault disappeared.
- Reconnect the PCB to the μC.
- There seems to be some trouble with the temperature sensor, it wont respond. there is no voltage at SCL and SDA pin, it should be pulled high by nrf52. I think it would be better to connect the temperature sensor directly to the nRf52 and not to the 3v3 regulator, since there is potentially differences from the regulator to the nrf52 3v3 output.
- Remove the connection from the regulator to the LM75B and create a wire from VDD nrf52 to it.

- Measuring the SCL and SDA, it is both pulled high now but there is nothing that's being sent.
- The temperature sensor seems not to work. I will replace it with a new one.
- The new temperature sensor worked and the design seems to work.

Conclusion It would be beneficial to use the 12V to power the nRF52, since it need to run for a long time without battery and computer connected, the relay also need the voltage in order to trigger. by this solution we got a circuit that got it power from 230VAC and is transformed down to 12VDC, and then regulated down to both 3v3 and 9V. The temperature sensor LM75BD seems to need powering directly from the nRF52 in order to work.

References Hardware files for the NRF52 `https://www.nordicsemi.com/eng/nordic/download_resource/50980/4/5171612/93935`

Figures & Graphs The schematics is placed in the appendix and is found at:
Relay circuit: [7.6.3].

3V3 regulator circuit: [7.6.3].

9V regulator circuit: [7.6.3].

7.5.4 Testing of the LT1072 - Buck boost converter circuit

Sondre Håverstad
Date: 12.04.17

The purpose of this test i to check of its possible to create a buck boost converter that boost 12V to about above 30V, so it can be used to power the Meter-bus without a external power supply. For testing the circuit, it will be constructed on a breadboard. This circuit is chosen because it is familiar to us from previous work. We know that it works and should be an easy circuit to construct.

Materials

- Electrolyte capacitors: 47uF and 470uF.
- Ceramic capacitor: 3nF.
- LT1072.
- Schottky barrier diode: MBRS340.
- Resistors: 200, 1k, 2k, 52k.
- Inductor: 150uH.
- Power supply with 12Vdc output.
- A multimeter or oscilloscope to do measurements with.
- Simple M-BUS circuit to test the buck boost converter against [7.5.1].

Methods

- Draw the circuit in LTspice to check if the values the current output is correct, circuitry shown in figure 72.
- Output if the simulation i shown in figure 74a.
- Construct the circuit on the breadboard by following the circuit in figure 72.
- Apply 12 Vdc to the input of the circuit.
- use the oscilloscope to measure the output, mine output shown in figure 73b.
- Connect the buck boost to the simple M-BUS converter by connecting the output of the buck boost to MBUS+ on the M-BUS converter.
- The output of the buck boost converter falls to 17.89V, this is a problem, because the simple M-BUS converter need more voltage to transmit the message through M-BUS.
- I reduced resistor R1 to 26kΩ and R2 to 1kΩ , to check if R2 had to high resistance.
- measured the output after connecting the simple M-BUS transceiver, got the same output.
- Got help from a supervisor and it was two pins called E1 and E2 at the LT1072 that wasn't connected to ground, causing the voltage to drop when connected to a load.
- Connect pin E1 and E2 to Ground.
- Connect the buck boost to the simple M-BUS converter, and measure the voltage, it should be as expected, as shown in figure 74b, the ripple voltage shown in figures: 75a and 75b.

Conclusion The Buck boost converter works fine after connecting E1 and E2 to ground. It shouldn't have problem powering the simple M-BUS converter. A thing to have in mind to later project is to connect all unconnected pins to something otherwise you're told not to.

References Datasheet and information about the LT1072:
`http://www.linear.com/product/LT1072`.

Figures & Graphs Simulation results and circuitry.

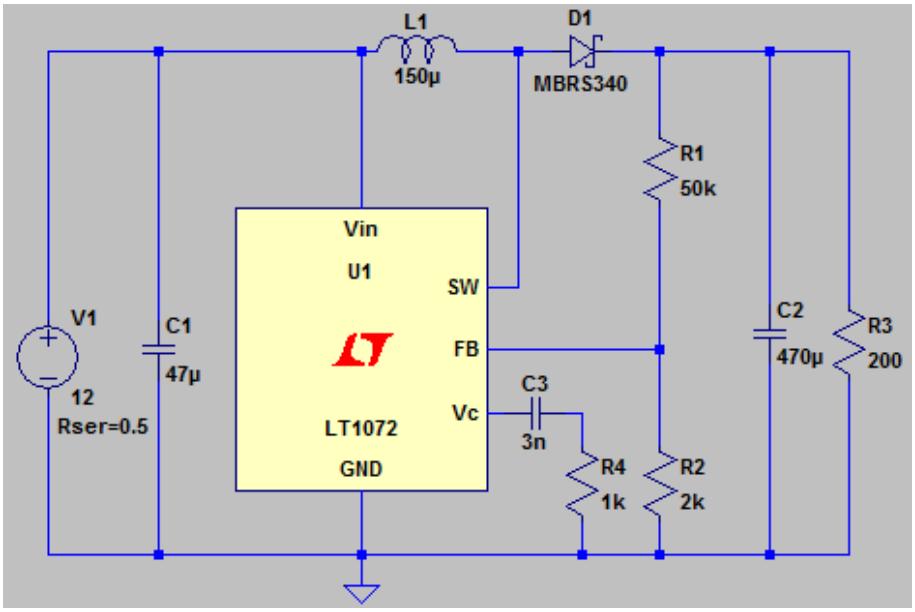
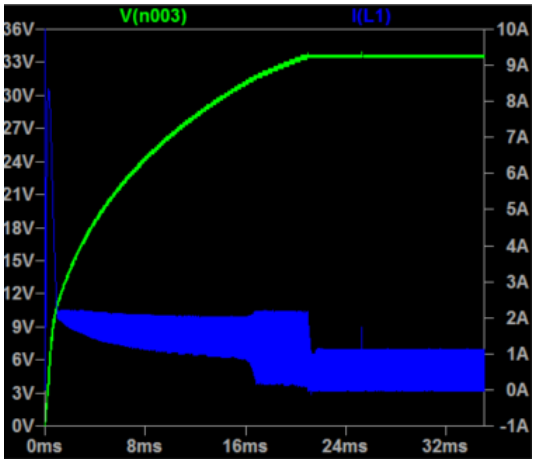
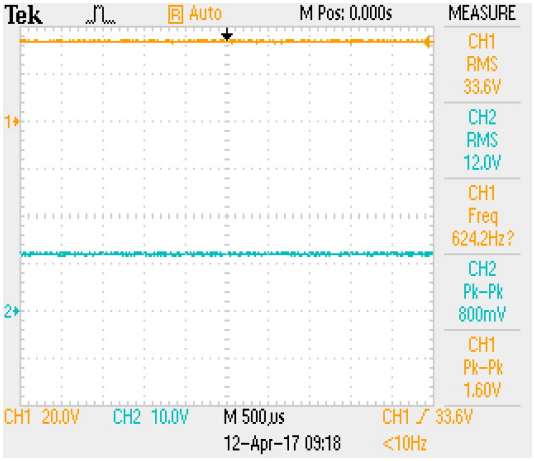


Figure 72: Buck boost simulation circuit



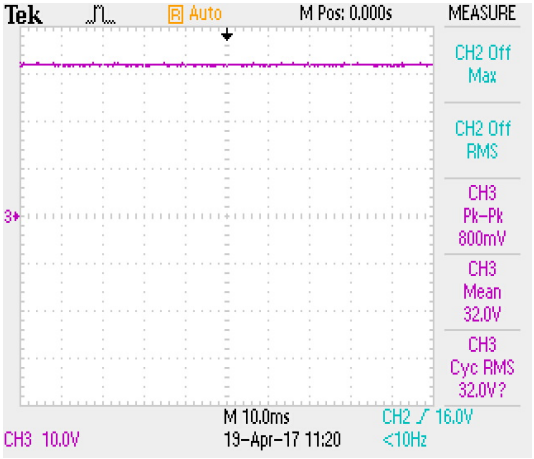
(a) Simulation result of the buck boost converter



(b) Measurement of the buck boost converter

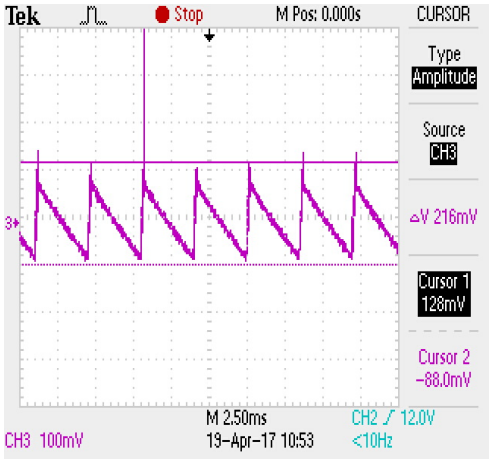
Figure 73: Measurement buck boost 1

(a) Simulation results the buck boost output

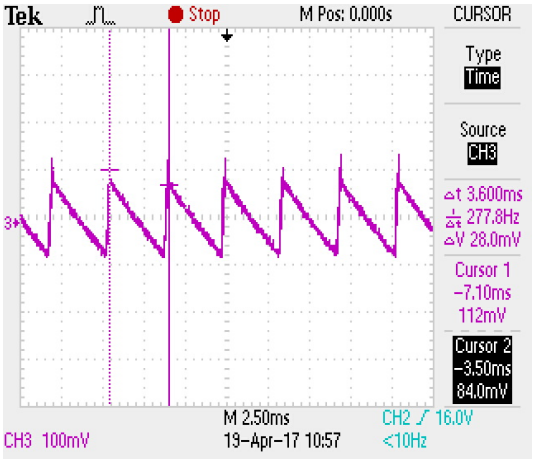


(b) Measurement buck boost output

Figure 74: Measurement buck boost 2



(a) Measurement ripple voltage



(b) Measurement ripple Peak-to-Peak

Figure 75: Measurement buck boost 3

7.5.5 Testing of the complete master circuit

Sondre Håverstad
Date: 26.04.2017

Introduction/Purpose The master device is a merged version of previous testing and will contain 12V to 3V3 regulator the test report: [7.5.2], the simple M-BUS transceiver [7.5.1] to communicate with watt-meter with M-BUS interface and a buck boost converter, which purpose is to power the the meter bus, [7.5.4]. This report is more like a system test, that the whole system works together.

Materials

- Power-supply with 12V output with probes.
- Multimeter.
- Watt meter with M-BUS interface to test simple M-BUS transceiver against.

Methods

- Mount the PCB on the master controller
- Connect 12Vdc adapter to the power jack plug.
- The red led diode lightens up down in the left corner.
- Measured regulator output and it was 2,8V, then changed the 470Ωresistor to 561Ω, the result was 3.1V, which is acceptable.
- Measured the output at the lower pin at P22, shown in figure 76. The output got measured to just above 27V, adjusted resistor R61 to 36KΩ, circuit shown in the schematics [7.5.7], resulting in a 32V which is sufficient.
- 3V3 regulator and the buck boost converter works, its time to test the transceiver.
- figure 76 contains measuring points 1, 2, 3.
- Point 3 is the supply line of the M-BUS, and is connected to the Buck boost, it should hold the same voltage. measured to 31.5V.
- The voltage at measuring point 1 should be close to 3V, it is the UART RX pin on the NRF52 and is pulled up. I measure the voltage = 2.8V.
- At measuring point 2. should hold around 12 volts and is the voltage divided which transmits the signal on MBUS-. the Voltage at point 2. is 11.5V.
- All the voltages in the simple M-BUS transceiver circuit seems to be correct, but it needs to be tested together with the software in another test.
- The transceiver for the AMS smart meter cannot be tested because it is not available.

Conclusion All the the parts of the master design has been individually tested except for the M-BUS transceiver for the AMS smart meter, its made from the documentations but which will not be tested in this thesis, because it is not possible. But the hardware needs to be tested against software to ensure that is function properly, test will be found in the appendix at [7.5.6].

References Buck boost data sheet: [45].
Regulator data sheet: [42].
Simple M-BUS circuit, found at: `https://github.com/rscada/libmbus/blob/master/hardware/MBus_USB.pdf`.

Figures & Graphs Figure 76 shows transceivers measuring points.

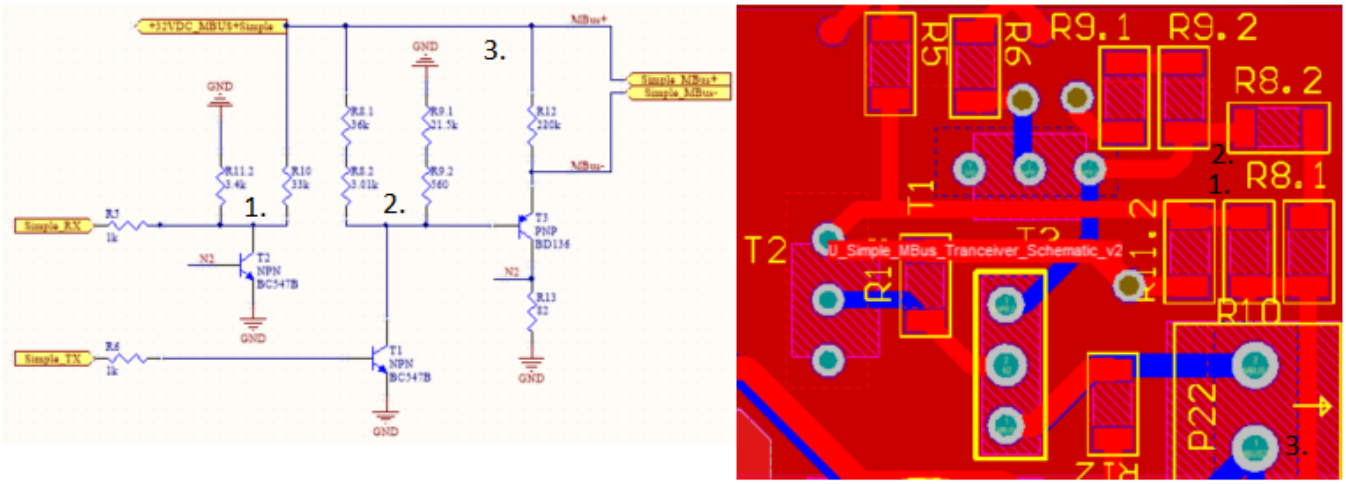


Figure 76: Simple M-BUS transceiver measuring points

7.5.6 Test master software

Eivind Stendal
Date:28.04.2017

Introduction/Purpose The purpose of this test is to check if master BLE protocol and controller task is doing what it should do.

Materials

- Multimeter
- Phone/ app
- Putty

Methods

- I made a checklist over all functions the master should have.

Data

Test sw master	Date: 28.04.2017 Tester: Eivind Stendal	
Task	**Comment**	**Checklist**
Master as peripheral, BLE connections with phone		
App: Set values to power-limits		ok
App: Set values to wanted temperature on slaves		ok
App: Set values to clock		ok
App: Set values to max power on slaves.		ok
App: Get a response from master after values are sat.		ok
App: If the NUS message is correct you should receive the new value.	Fix clock received msg to xx:00	
App: If the NUS message is not correct you should receive a error message	You dont get an error if you write a letter instead of a number	ok
Restarts after powerloss		
Master as central, BLE connections with slaves		
General		
Can connect with up to 7 slaves.	only tested with 2 slaves but it should work	
Stops sanning when connected with 7 slaves.	not tried becouse we only got 2 slaves.	
Still scanning when connected to one slave		ok
Restarts after powerloss		ok
Sending to slave(n) when		
If the master receives a msg from a slave with no address it should respond with a address.		ok
A slave is supposed to turn on		ok
Ack timer runs out		ok
A new wanted temp is received from app.		ok
Ack		
The master should resend datapacket if it get lost.		ok
If the master receives a packet that is not a ack it shoudl respond with an ack msg.		ok
It only resends a lost packet once.		ok
Controller		
Power consumption received in controller task.	wrong value printed 1400 should be 60.(Fixed,RTOS problem)	ok
The controller starts to turn off slaves in low priority if we are exceeding preferred_consume_limit		ok
The controller starts to turn off slaves in low priority if we in "high price hours".		ok
The controller starts to turn off slaves in normal priority if we are exceeding normal_consume_limit		ok
The controller starts to turn off slaves in High priority if we are exceeding max_consume_limit	only for slaves with high priority	ok
Every time a slave is turned of the slave_on_timer is started/ restarted.		ok
When the slave_on_timer runs out the the slave with the highest priority will be turned on if we are using less power then preffered consume limit.		ok
Cont from above; If there is more slaves with same priority the slave with the highest temperature/ wanted temp deviation.		ok

Buttons and leds		
Led 1 blinking: Scanning		**ok**
Led 1 on: Connected to phone		**ok**
Buttons 2: Start adverticing to connect to app		**ok**
Known weaknesses		
If an android device is disconnected only with the disconnect button on app the device reconnect and disconnects untill the bluetooth is turned off, even if the NRF52 is restarted. This seems like an application weakness.		
If conneced with phone and restarts slave the program chashes???	fixed	**ok**
I didnt receiv any temp from slave?? This happend once, but it sends if there is change in temp	The slave is now programmed to send temperature after the first update from sensor.	**ok**
com error 1,1 -> Overrun error	Fixed by not letting ble interrupts run before uart scan is done.	**ok**
If to slaves tries to connect at the same time only one will succeed.	Fixed by resetting slave after 10 seconds if no address is given.	
Two slaves got the same address if turned on at the same time.	pointer mistake, fixed	**ok**
Master doesnt work if the usb is not connected	The master had a current sensing resistor from a earlier project that made the voltage on the nrf to low. We fixed this by removing the resitor.	**ok**
com error 1,1 -> Overrun error,triggered after 10min, the program crashed and restarted 10 minutes later		
Com error 4,4		

Figure 77: Checklist for software functionality on master

Conclusion The system is working good, however we get a com error at times, that is triggered by the UART/m-bus thread. It seems like this is triggered by wrong power level on RX pin, when the M-Bus is not powered.

7.5.7 Test slave software

Eivind Stendal and Jan Roar Mydland
Date:26.04.2017

Introduction/Purpose The purpose of this test is to check if slave device is doing what it should do, based on the commands from the master. And to test that TWI work as expected.

Materials

- Multimeter
- Power supply 12V
- Putty
- Salea logic analyzer with related computer software

Methods

- I (Eivind) made a checklist over all functions the slave should have. See 78.
- Connected up the Salea logic analyzer on the SCL and SDA pin regarding the TWI.
- Figure 79 shows the request temperature sending and the response from the temperature sensor.
- On the blue lines in channel 1 and 2 on the figure 79 shows a analog view of the data and clock lines. The curves seems like it is a bit slow to get to a logic high.

Test sw slave	Date: 26.04.2017 Tester: Eivind Stendal	
Task	**Comment**	**Checklist**
Slave ble protocoll		
Turn on or off relay based on signal from master		**ok**
Sends when		
slave is not given any address	timer	**ok**
there is a change in temperature 1 degree celcius or more	timer	**ok**
not received any ack for sent packet. Only once		**ok**
Thermostate		
Temperature is calibrated	Eksternal sensor would be better	**x**
Temperature is stored in a variable		**ok**
Slave is initialized in low priority		**ok**
If temperature is 2 degrees or more colder then wanted temperature the priority is set to normal.		**ok**
When changing priority the master is updated		**ok**
Relay is turned on if wanted temp is higher then current temp and the master allows it to turn on		**ok**
Leds and buttons		
Led 1: Advertising		**ok**
Led 3: state on/off Allowed to turn on by master		**ok**
Led 4: Relay on/off		**ok**

Figure 78: Checklist for functionality on slave

Data

Conclusion All the functions is working good, however the on-board temperature sensor is not optimal because of the heat from the microcontroller. For a functional system an external sensor would be necessary. The sending over TWI works fine, but it can seem like the gpio driver strenght could be set a little higher or the frequency of the TWI a bit slower in order to get a faster logic high level.

References More information on the Salea logic analyzer can be found here:
https://www.saleae.com/

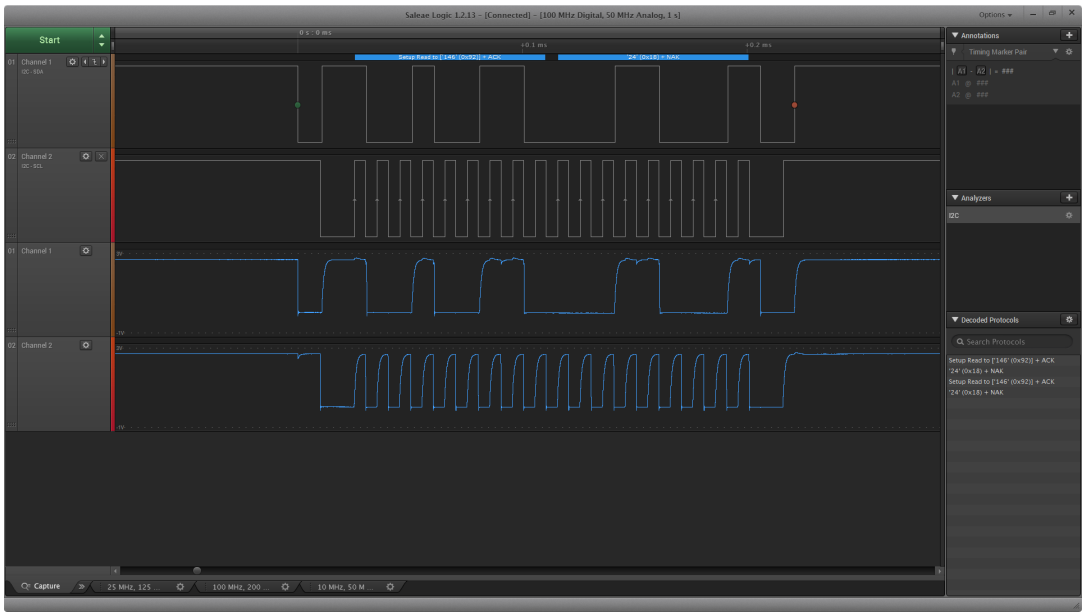


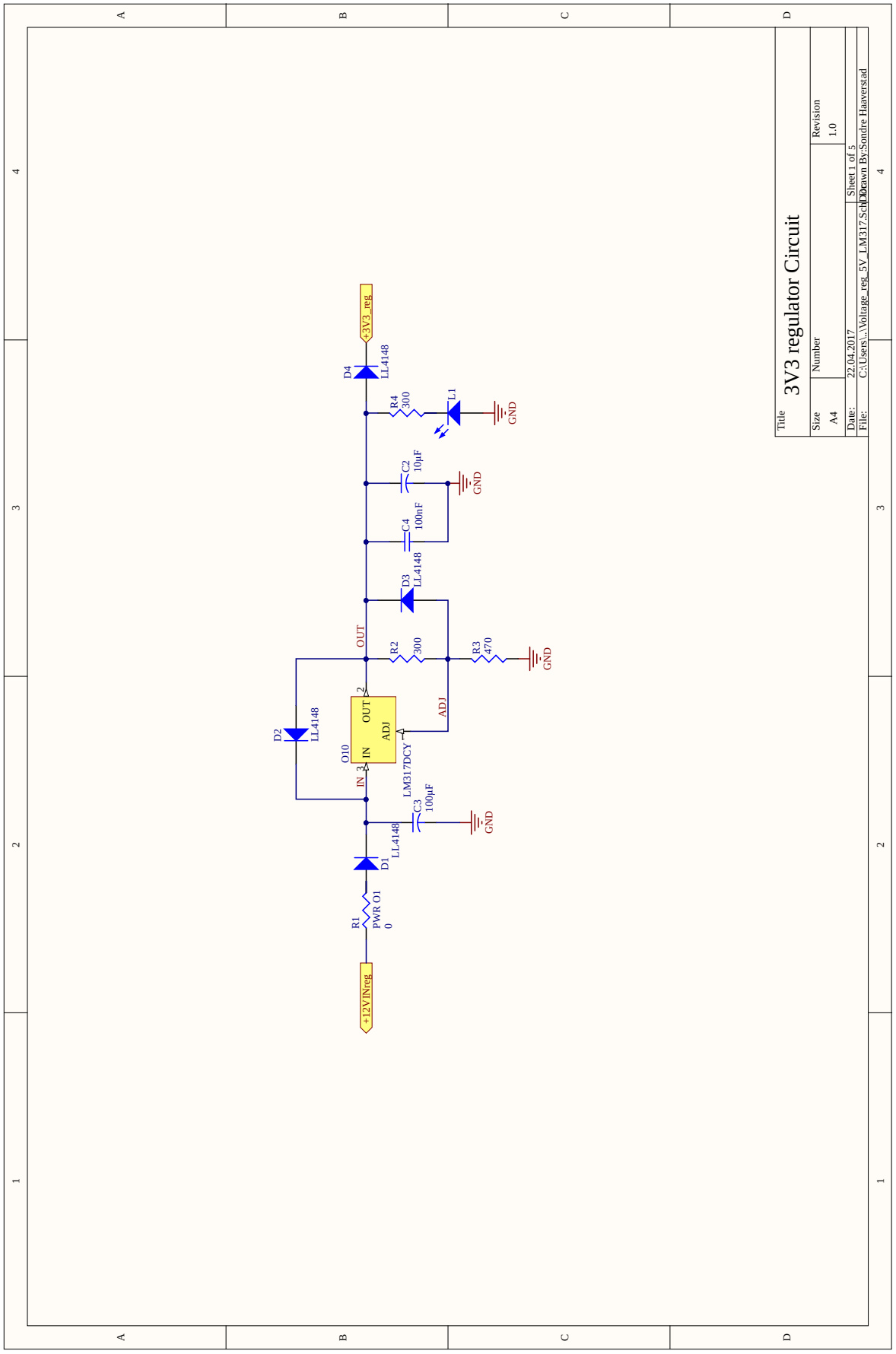
Figure 79: Sending request and receiving temperature as seen on the logic analyzer

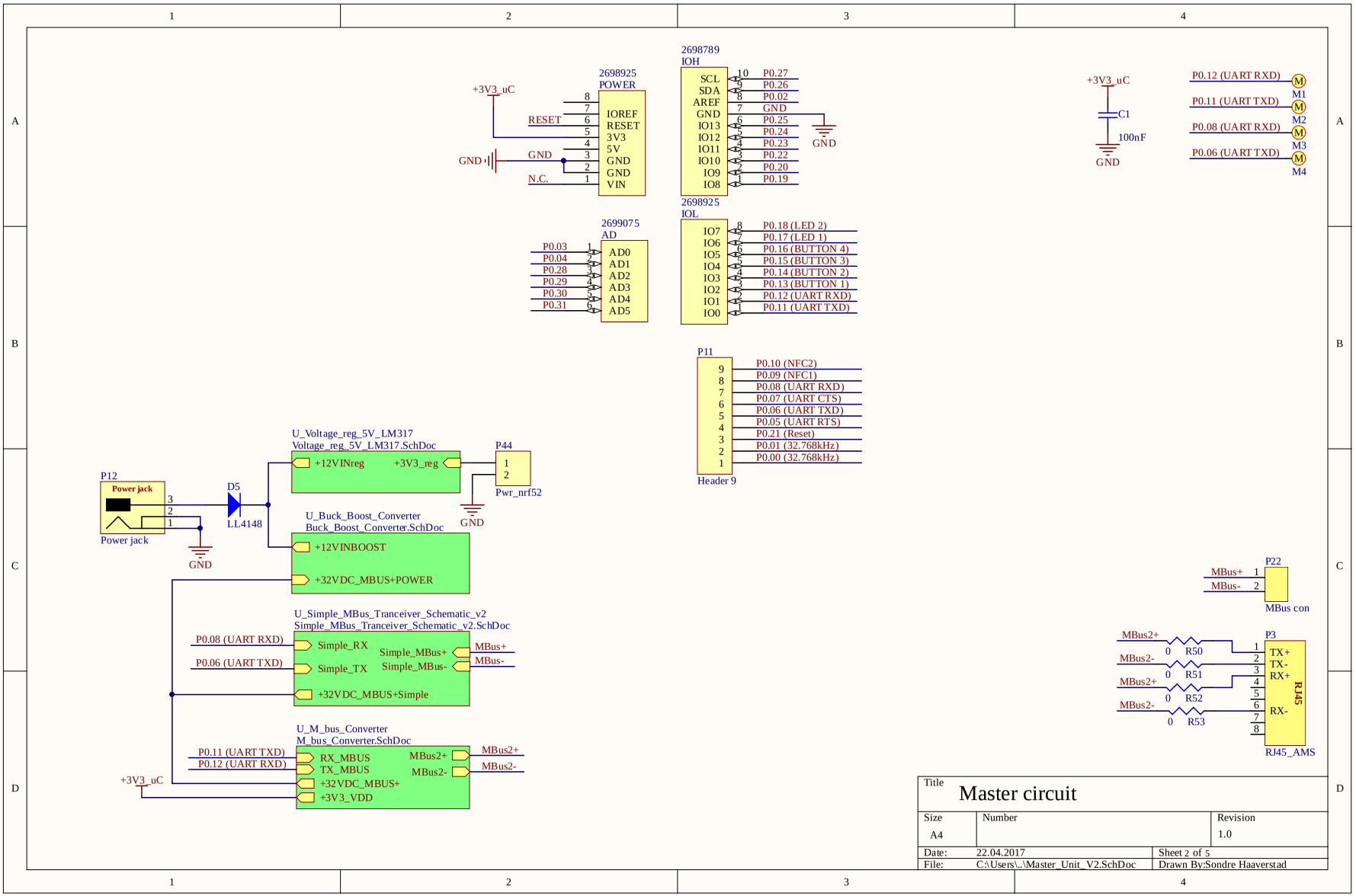
7.6 Appendix E - Hardware schematics

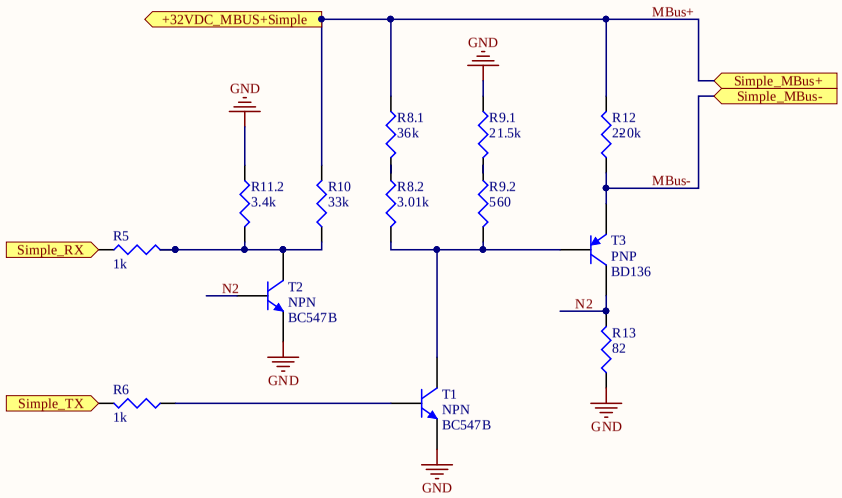
7.6.1 BOM Master

BOM master device					
Comment	Description	Designator	Quantity	Mounting Type	Value
AD	Conncetor Arduino shield	AD	1	Through Hole	
CAP, SSC, Avknrf52	Capacitor Ceramic	C1, C4, C9	3	Surface Mount	100nF
Cap 10uF	CAP TANT 10UF 16V 10% 1411	C2	1	Surface Mount	10µF
ECA-1EHG101	CAP ALUM 100UF 20% 25V RADIAL	C3	1	Through Hole	100µF
STC	Polarized Capacitor (Radial)	C5	1	Through Hole	200uF
Cvdd	Capacitor Ceramic	C6	1	Surface Mount	100n
Csc x2	Capacitor Ceramic	C7, C8	2	Surface Mount	330pF
Cap Ceramic	Capacitor Ceramic	C11	1	Surface Mount	1n
ECE-A1CN470U	CAP ALUM 47UF 20% 16V RADIAL	C15	1	Through Hole	47µF
EEE-FK1H471AM	Electrolytic Capacitors - SMD 470UF 50V FK SMD	C17	1	Through Hole	470 uF
LL4148	DIODE GEN PURP 100V 200MA SOD80	D1, D2, D3, D4, D5	5	Surface Mount	
NTE573	DIODE	D6	1	Through Hole	
SRR1260A-151K	FIXED IND 150UH 1.55A 260 MOHM	I1	1	Surface Mount	
IOH	Conncetor Arduino shield	IOH	1	Through Hole	
IOL	Conncetor Arduino shield	IOL	1	Through Hole	
LED red through hole	Through Hole Red LED	L1	1	Through Hole	
TSS721ADR	IC TXRX METER BUS 16SOIC	O1	1	Surface Mount	
LT1072	IC REG MULT CONFIG INV ADJ 8SOIC	O2	1	Surface Mount	
LM317DCY	IC REG LINEAR ADJ 1.5A SOT223-4	O10	1	Surface Mount	
RJ45_AMS	RJ45	P3	1	Through Hole	
Header 9	Header, 9-Pin	P11	1	Through Hole	
Power jack	Pwr connector Jack 9mm	P12	1	Through Hole	
MBus con	2-Way screw connector	P22	1	Through Hole	
Pwr_nrf52	Header, 2-Pin	P44	1	Through Hole	
POWER_UNO	Conncetor Arduino shield	POWER	1	Through Hole	
RES SMD 1206	RES SMD 1.15K OHM 1% 1/4W 1206	R3	1	Surface Mount	470
RC1206JR-07300RL	RES SMD 300 OHM 5% 1/4W 1206	R2, R4	2	Surface Mount	300
ERJ-8ENF1001V	RES SMD 1K OHM 1% 1/4W 1206	R5, R6	2	Surface Mount	1k
ERJ-8ENF3602V	RES SMD 36K OHM 1% 1/4W 1206	R8.1	1	Surface Mount	36k
ERJ-8ENF3011V	RES SMD 3.01K OHM 1% 1/4W 1206	R8.2	1	Surface Mount	3.01k
ERJ-8ENF2152V	RES SMD 21.5K OHM 1% 1/4W 1206	R9.1	1	Surface Mount	21.5k
RC1206JR-07560RL	RES SMD 560 OHM 5% 1/4W 1206	R9.2	1	Surface Mount	560
RC1206FR-0733KL	RES SMD 33K OHM 1% 1/4W 1206	R10	1	Surface Mount	33k
ERJ-8ENF3401V	RES SMD 3.4K OHM 1% 1/4W 1206	R11.2	1	Surface Mount	3.4k
RC1206FR-07220KL	RES SMD 220K OHM 1% 1/4W 1206	R12	1	Surface Mount	220k
RC1206JR-0782RL	RES SMD 82 OHM 5% 1/4W 1206	R13	1	Surface Mount	82
Res3	Resistor	R14, R15	2	Surface Mount	220
Rris	Resistor	R16	1	Surface Mount	330
RIDD	Resistor	R17	1	Surface Mount	30k
Rload	Resistor	R18	1	Surface Mount	100k
resistor, PWR O1	Resistor	R1, R50, R51, R52, R53	5	Surface Mount	0
ERJ-8ENF1001V	RES SMD 1K OHM 1% 1/4W 1206	R60, R62	2	Surface Mount	1k
RC1206FR-0724K3L	RES SMD 24.3K OHM 1% 1/4W 1206	R61	1	Surface Mount	24.3k
BC547B	TRANS NPN 45V 0.1A TO-92	T1, T2	2	Through Hole	
BD136	TRANS PNP 45V 1.5A SOT-32	T3	1	Through Hole	

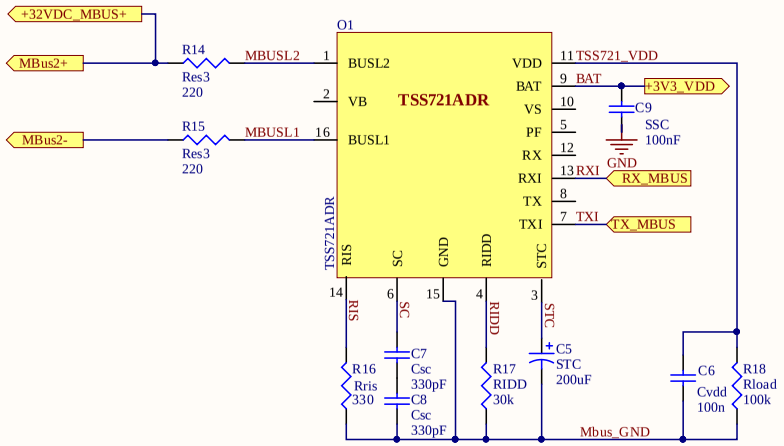
7.6.2 Master schematics & overview



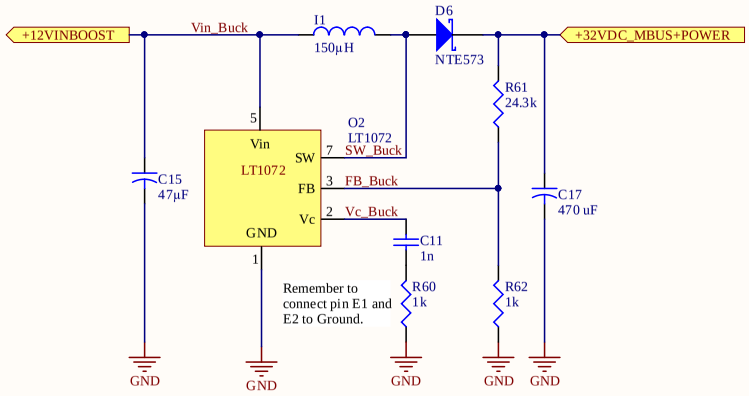

2698925
POWER
+3V3_uC
RESET
GND
N.C.
IOREF
RESET
3V3
5V
GND
GND
VIN
2698789
IOH
SCL
SDA
AREF
GND
IO13
IO12
IO11
IO10
IO9
IO8
P0.27
P0.26
P0.02
GND
P0.25
P0.24
P0.23
P0.22
P0.20
P0.19
2698925
IOL
IO7
IO6
IO5
IO4
IO3
IO2
IO1
IO0
P0.18 (LED 2)
P0.17 (LED 1)
P0.16 (BUTTON 4)
P0.15 (BUTTON 3)
P0.14 (BUTTON 2)
P0.13 (BUTTON 1)
P0.12 (UART RXD)
P0.11 (UART TXD)
2699075
AD
P0.03
P0.04
P0.28
P0.29
P0.30
P0.31
AD0
AD1
AD2
AD3
AD4
AD5
+3V3_uC
C1
100nF
GND
P0.12 (UART RXD)
P0.11 (UART TXD)
P0.08 (UART RXD)
P0.06 (UART TXD)
M1
M2
M3
M4
P11
P0.10 (NFC2)
P0.09 (NFC1)
P0.08 (UART RXD)
P0.07 (UART CTS)
P0.06 (UART TXD)
P0.05 (UART RTS)
P0.21 (Reset)
P0.01 (32.768kHz)
P0.00 (32.768kHz)
Header 9
U_Voltage_reg_5V_LM317
Voltage_reg_5V_LM317.SchDoc
+12VINreg
+3V3_reg
P44
Pwr_nrf52
P12
Power jack
D5
LL4148
U_Buck_Boost_Converter
Buck_Boost_Converter.SchDoc
+12VINBOOST
+32VDC_MBUS+POWER
U_Simple_MBus_Tranceiver_Schematic_v2
Simple_MBus_Tranceiver_Schematic_v2.SchDoc
P0.08 (UART RXD)
P0.06 (UART TXD)
Simple_RX
Simple_TX
Simple_MBus+
Simple_MBus-
MBus+
MBus-
+32VDC_MBUS+Simple
U_M_bus_Converter
M_bus_Converter.SchDoc
P0.11 (UART TXD)
P0.12 (UART RXD)
RX_MBUS
TX_MBUS
+32VDC_MBUS+
+3V3_VDD
MBus2+
MBus2-
P22
MBus con
P3
R50
R51
R52
R53
TX+
TX-
RX+
RX-
RJ45
RJ45_AMS
Title
Master circuit
Size
A4
Number
Revision
1.0
Date:
22.04.2017
Sheet 2 of 5
File:
C:\Users\..\Master_Unit_V2.SchDoc
Drawn By:Sondre Haaverstad



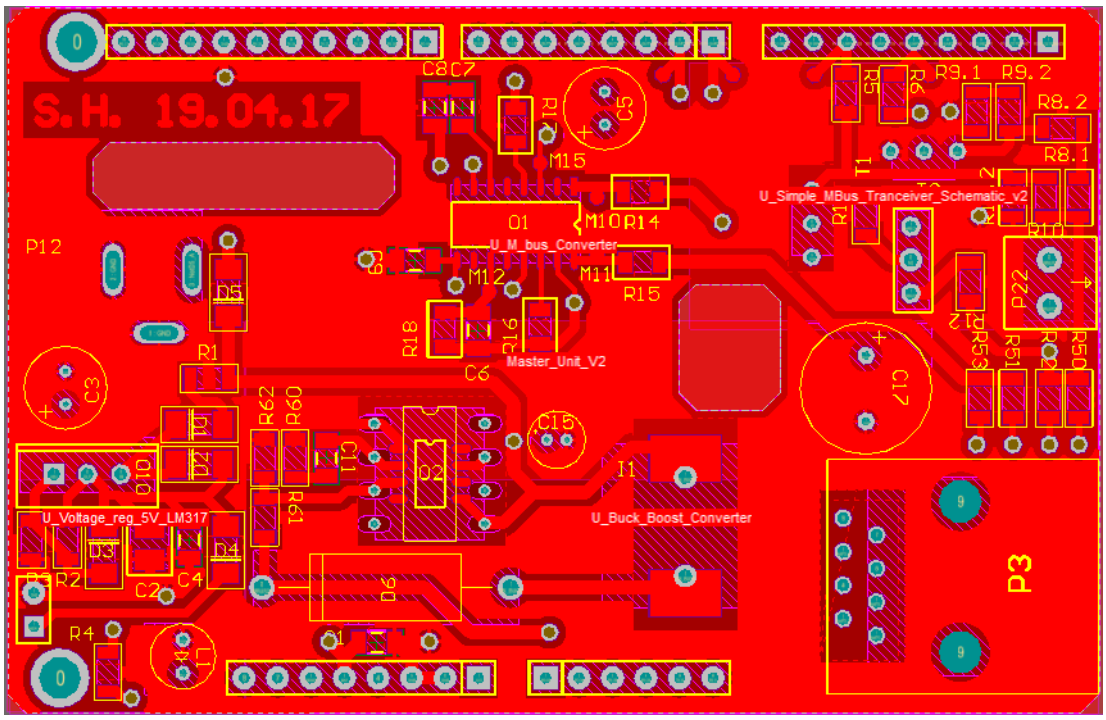
Title	Simple MBUS tranceivere	
Size: A4	Number:	Revision: 1.0
Date: 22.04.2017	Sheet 3 of 5	
File: C:\Users\...\Simple_MBus_Tranceiver_Schematic_V2.SchDoc	Drawn By: Jonas Haaverstad	

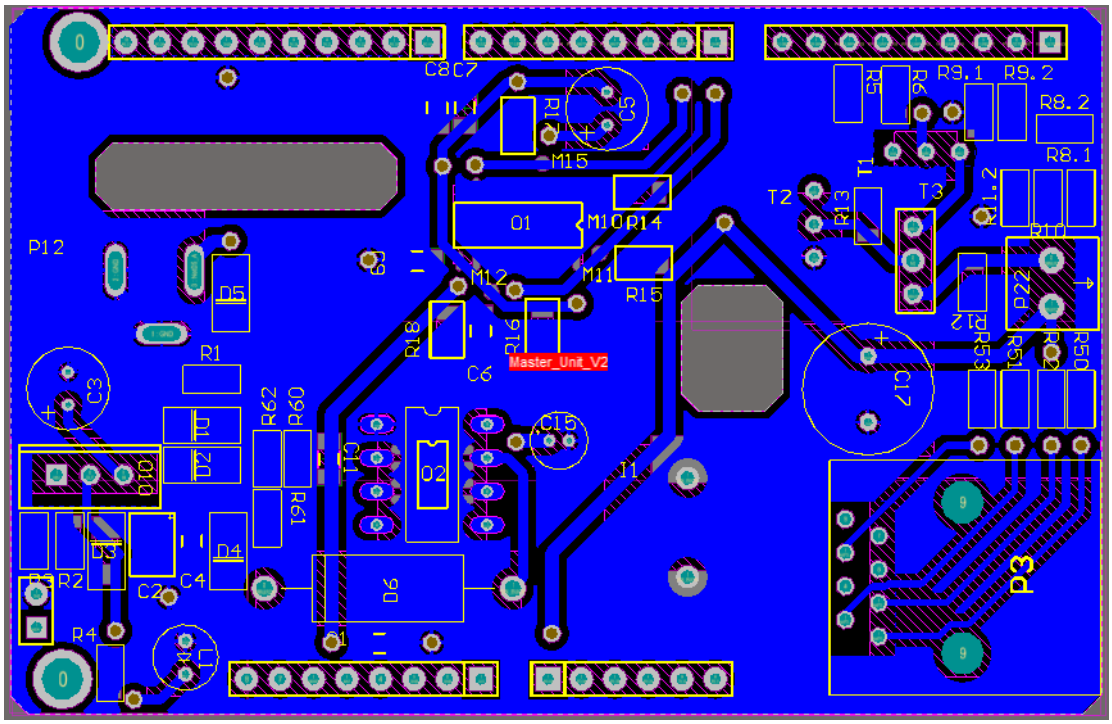


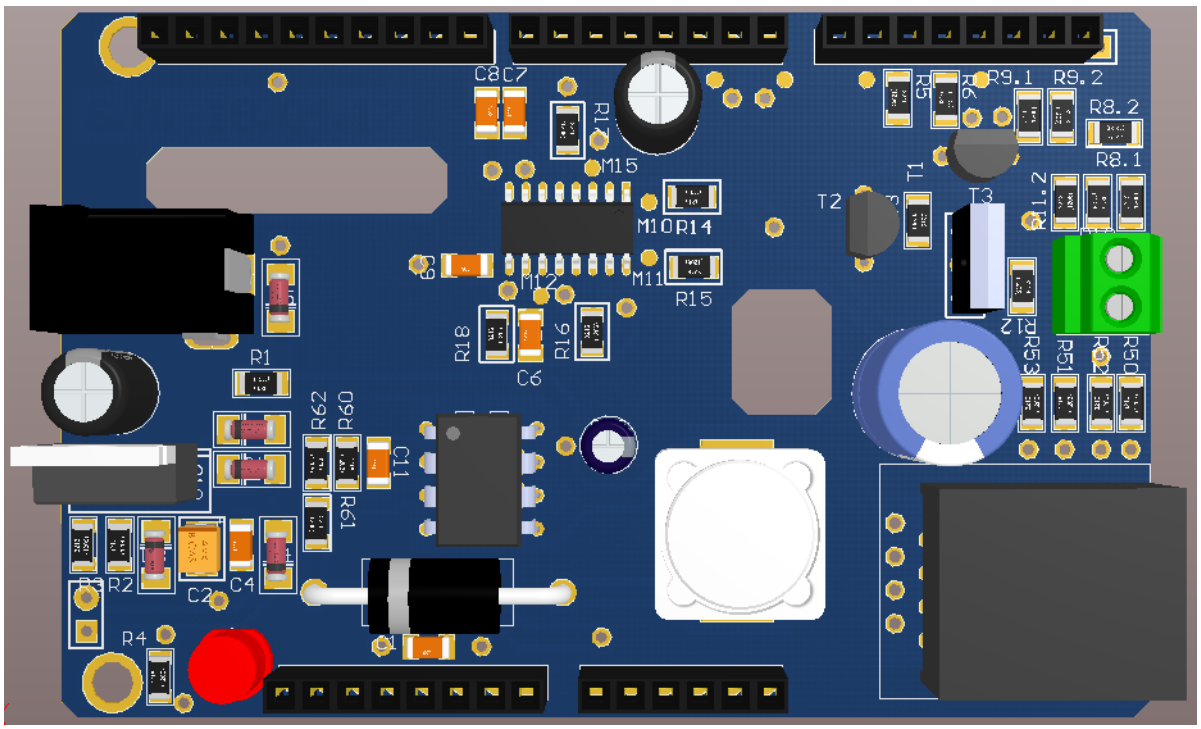
Title: Meter Bus Tranceiver		
Size: A4	Number:	Revision: 1.0
Date: 22.04.2017		Sheet 4 of 5
File: C:\Users\...\M_bus_Converter.SchDoc		Drawn By:Sondre Haaverstad

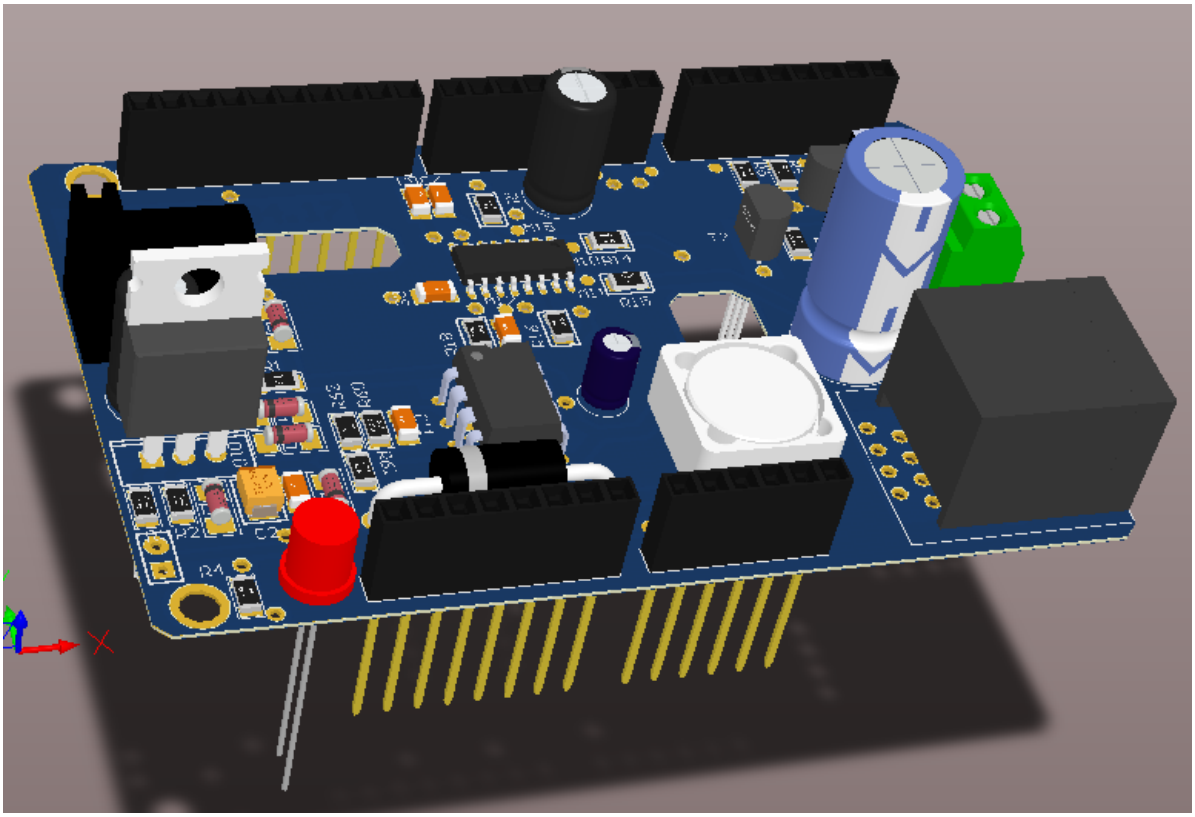


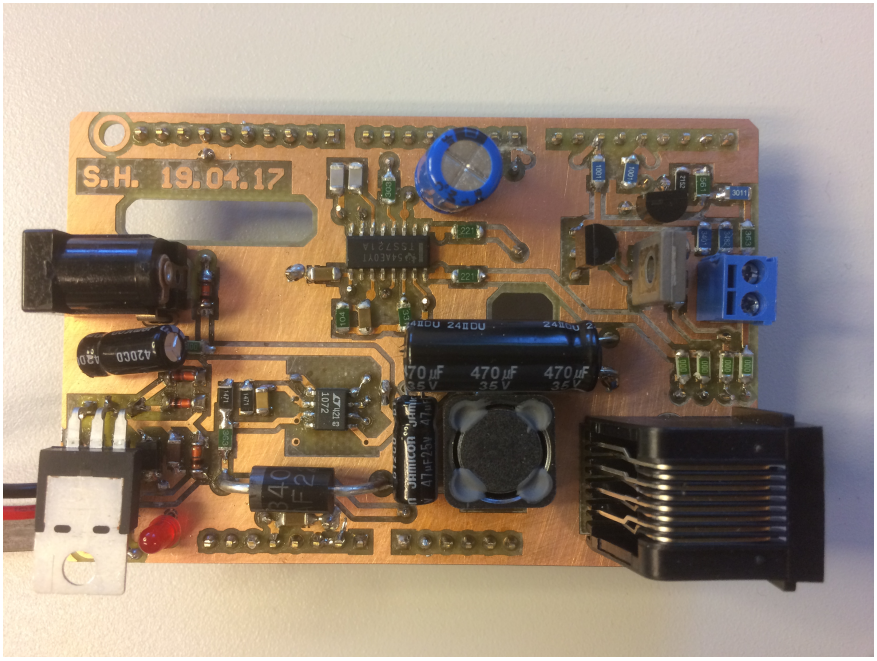
Title	Buck boost converter circuit		
Size: A4	Number:		Revision: 1.0
Date: 22.04.2017		Sheet 5 of 5	
File: C:\Users\..\Buck_Boost_Converter.SchDoc		Drawn By:Sondre Haaverstad	

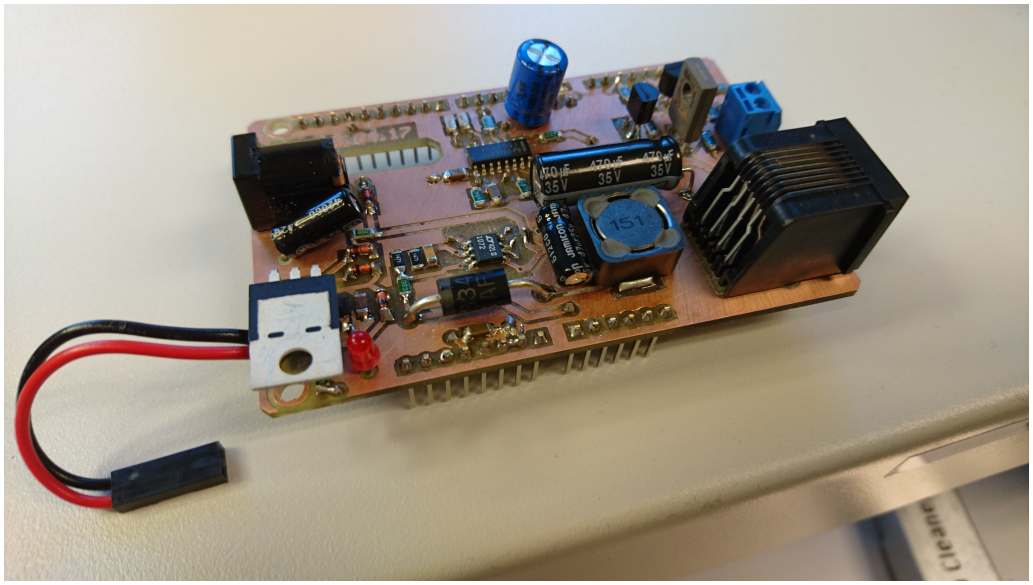

S.H. 19.04.17
U_M_bus_Converter
U_Simple_MBus_Tranceiver_Schematic_v2
Master_Unit_V2
U_Voltage_reg_5V_LM317
U_Buck_Boost_Converter
P12
P3


Master_Unit_V2
P12
P3


C8
C7
R1
M15
M10
R14
M12
M11
R15
R18
R16
C6
R1
R62
R60
C11
R61
R3
R2
C2
C4
R4
R9.1
R9.2
R8.2
R8.1
T1
T2
T3
R53
R51
R52
R50


R4
X



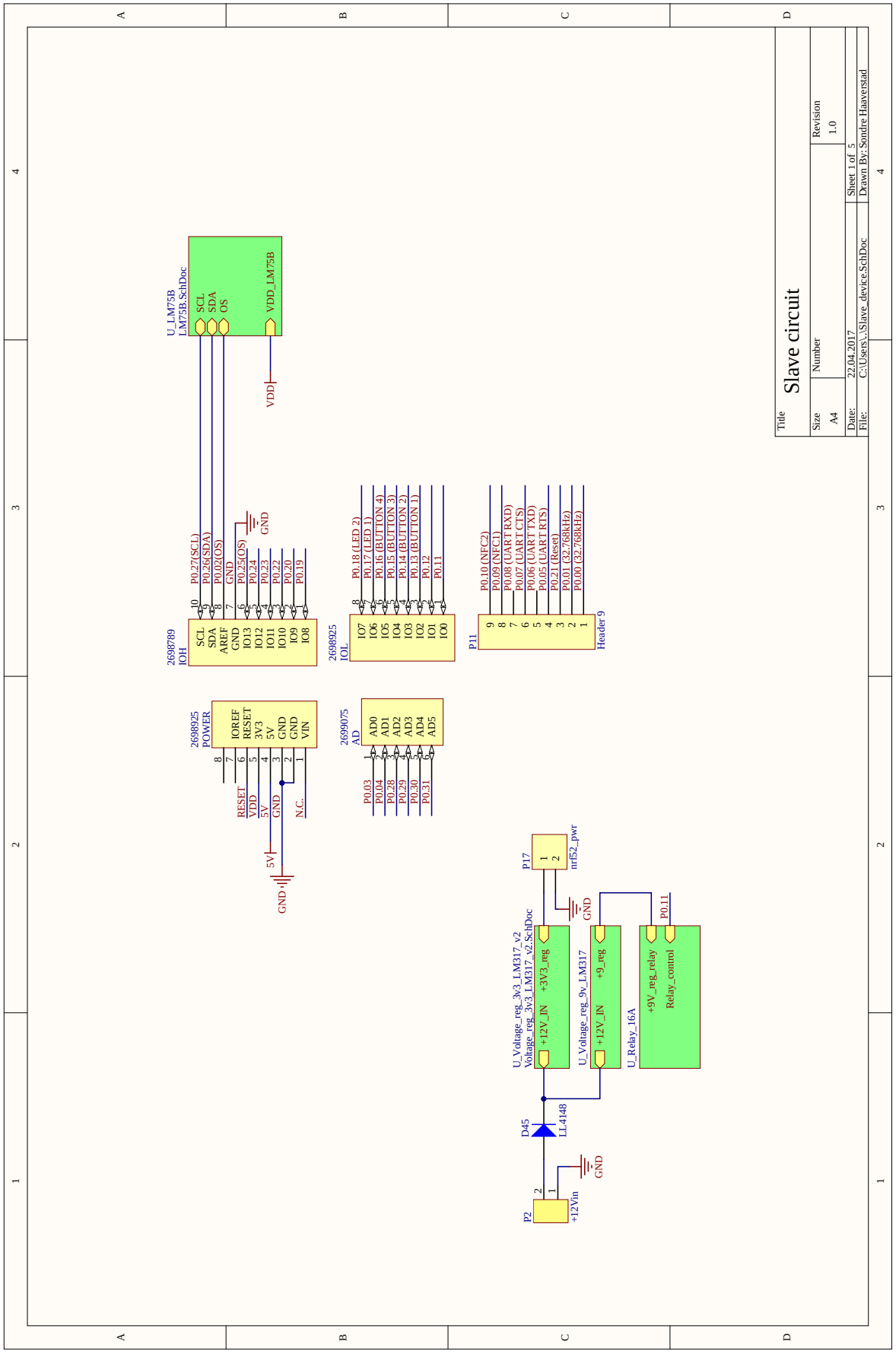


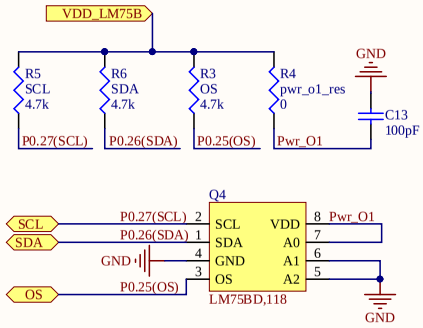
7.6.3 Result master PCB

7.6.4 BOM Slave

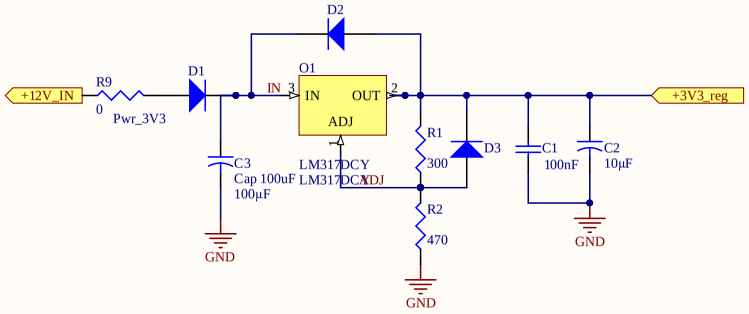
BOM slave device					
Comment	Description	Designator	Mounting Type	Quantity	Value
RC1206FR-071K5L	RES SMD 1.5K OHM 1% 1/4W 1206	R30	Surface Mount	1	1.5k
Cap 100uF	CAP TANT 100UF 16V 10% 2917	C3, C30	Surface Mount	2	100µF
Cap 10uF	CAP TANT 10UF 16V 10% 1411	C2, C20	Surface Mount	2	10µF
Cap	Capacitor ceramic 1206	C1, C10	Surface Mount	2	100nF
LM317DCY	IC REG LINEAR ADJ 1.5A SOT223-4	O1, O2	Surface Mount	2	
RC1206JR-07300RL	RES SMD 300 OHM 5% 1/4W 1206	R1, R10	Surface Mount	2	300
RC1206JR-07470RL	RES SMD 470 OHM 5% 1/4W 1206	R2, R20	Surface Mount	2	470
Diode 1N4002	DIODE GEN PURP 100V 200MA SOD80	D1, D2, D3, D7, D10, D20, D30	Through Hole	7	
Pwr_-3V3, -o1_res	Resistor	R9, R4, R14, R16	Surface Mount	4	0
+12Vin	2-Way screw connector	P2	Through Hole	1	
AD	Conncetor Arduino shield	AD	Through Hole	1	
Header 9	Header, 9-Pin	P11	Through Hole	1	
IOH	Conncetor Arduino shield	IOH	Through Hole	1	
IOL	Conncetor Arduino shield	IOL	Through Hole	1	
LL4148	DIODE GEN PURP 100V 200MA SOD80	D45	Surface Mount	1	
nrf52_pwr	Header, 2-Pin	P17	Through Hole	1	
POWER_UNO	Conncetor Arduino shield	POWER	Through Hole	1	
230V ON/OFF	MKDS 5/ 2-6,35	P1	Through Hole	1	
230VAC, 16A	Relay	Q2	Through Hole	1	
BC547B	TRANS NPN 45V 0.1A TO-92	T10	Through Hole	1	
ERJ-8ENF1001V	RES SMD 1K OHM 1% 1/4W 1206	R12	Surface Mount	1	1k
Avk LM75B	Capacitor ceramic 1206	C13	Surface Mount	1	100pF
LM75BD	LM75BD	Q4	Surface Mount	1	
OS, SCL, SDA	Resistor	R3, R5, R6	Surface Mount	4	4.7k

7.6.5 Slave schematics & overview

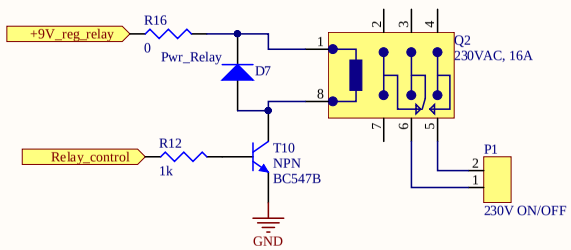




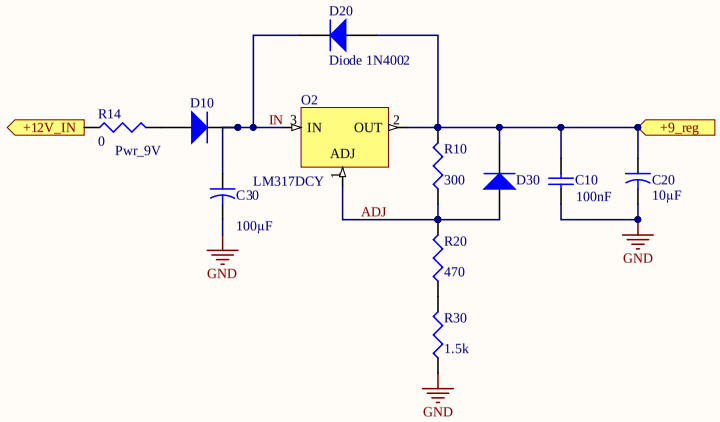
Title	Temperature sensor		
Size: A4	Number:		Revision: 1.0
Date: 22.04.2017			Sheet 2 of 5
File: C:\Users\..\LM75B.SchDoc			Drawn By: Sondre Haaverstad



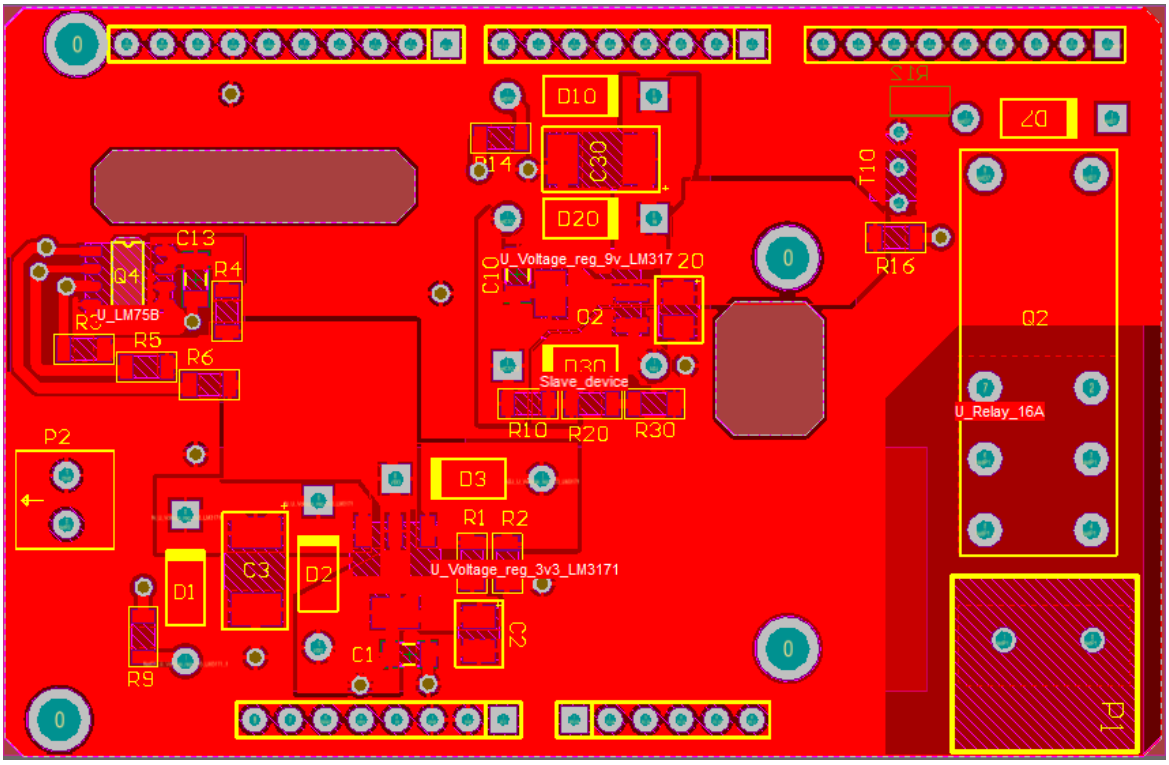
Title	3V3 regulator Circuit		
Size: A4	Number:		Revision: 1.0
Date: 22.04.2017		Sheet 3 of 5	
File: C:\Users\..\Voltage_reg_3v3_LM317_v2.SchDoc		Drawn By: Sondre Haaverstad	

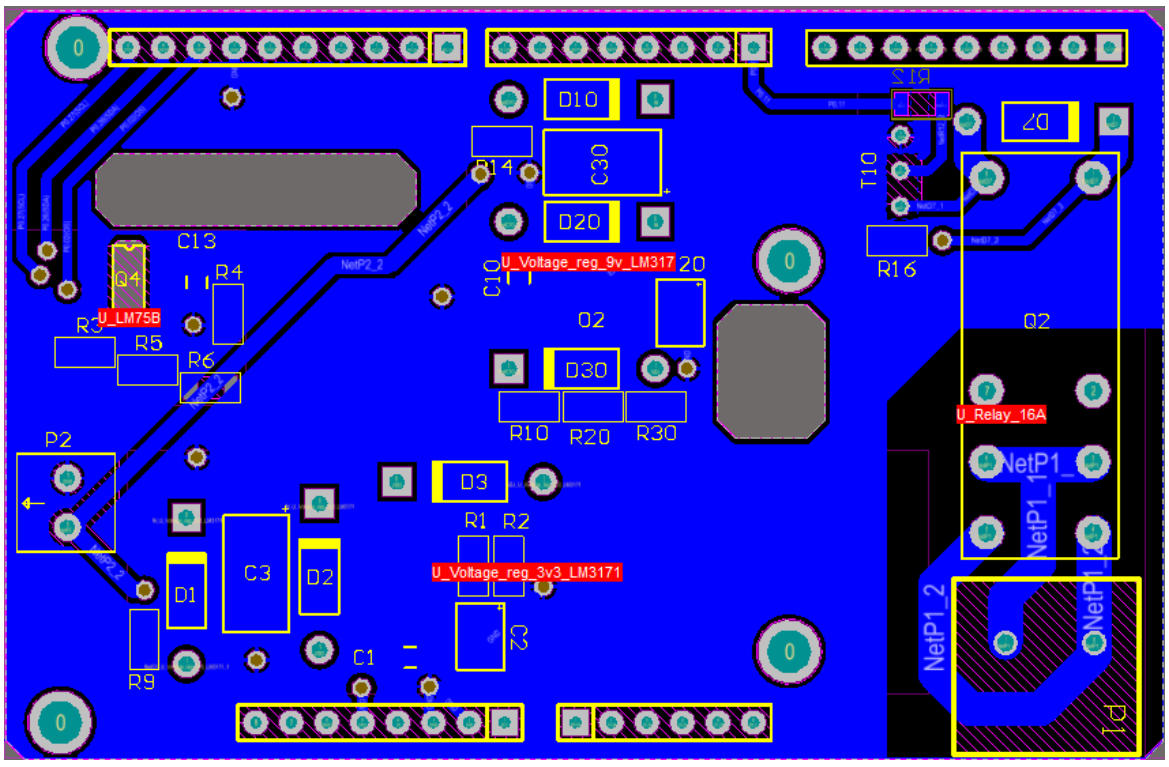


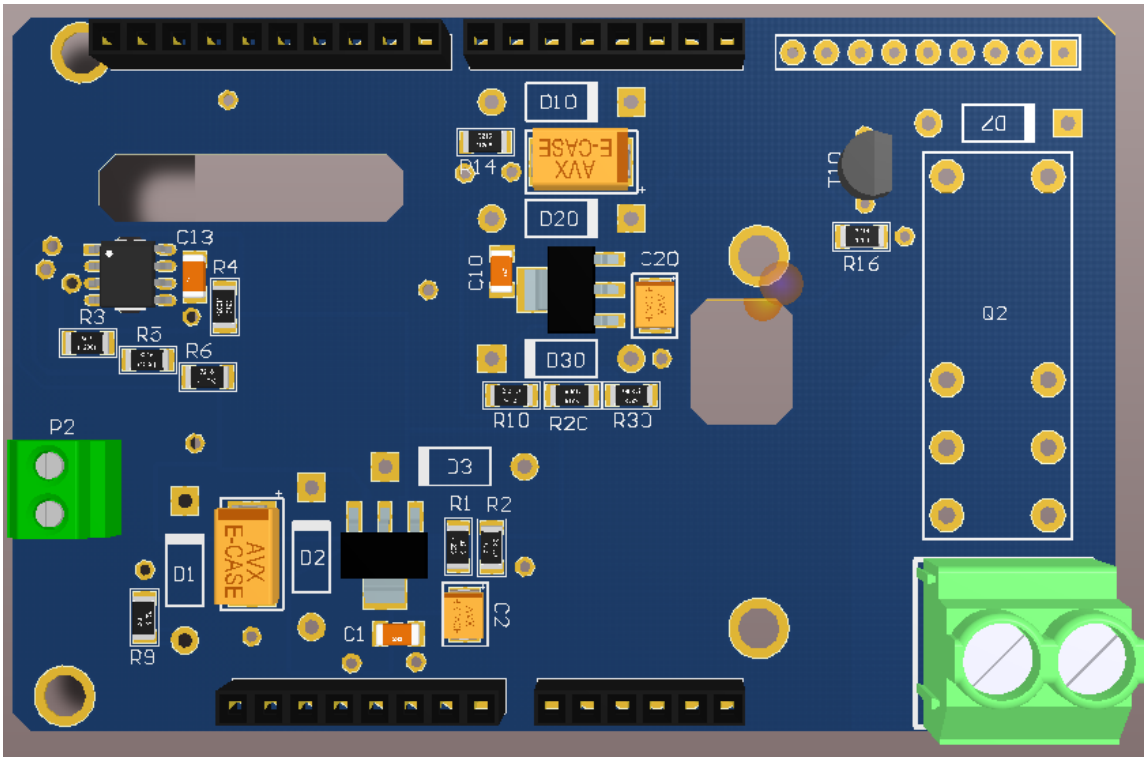
Title	Relay circuit		
Size: A4	Number:		Revision: 1.0
Date: 22.04.2017		Sheet 4 of 5	
File: C:\Users\..\Relay_16A.SchDoc		Drawn By: Sondre Haaverstad	

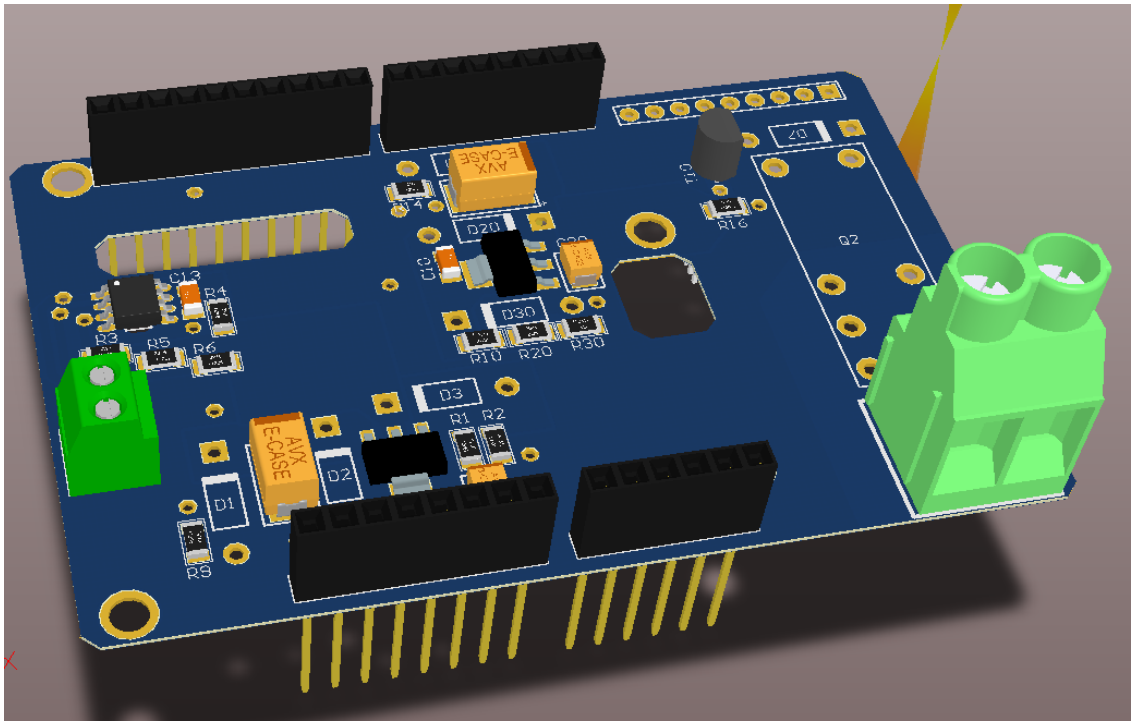


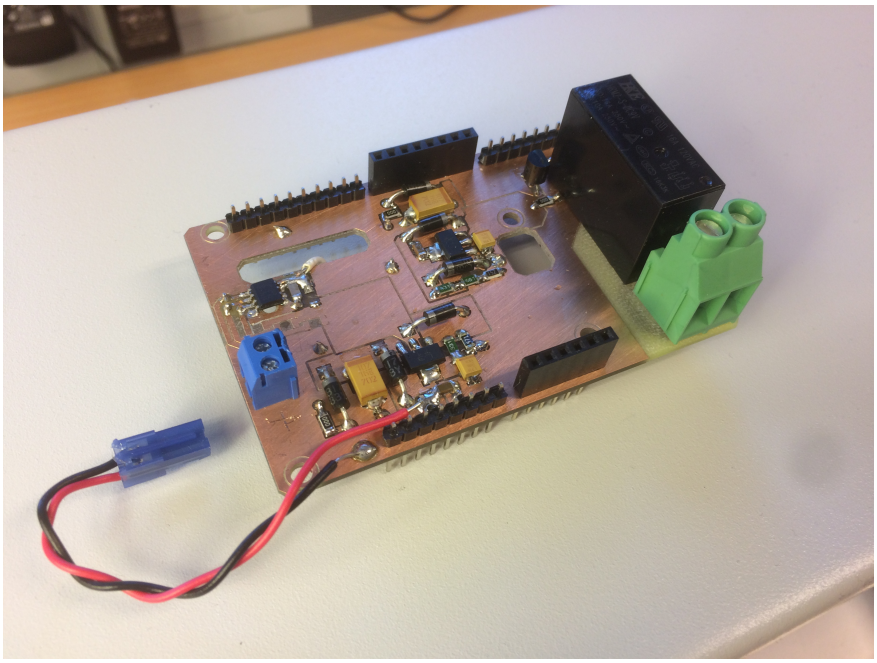
Title	9V regulator circuit		
Size: A4	Number:		Revision: 1.0
Date: 22.04.2017		Sheet 5 of 5	
File: C:\Users\..\Voltage_reg_9v_LM317.SchDoc		Drawn By: Sondre Haaverstad	


D10
C30
D20
R14
T10
D7
C13
Q4
R4
U_LM75B
R3
R5
R6
C10
U_Voltage_reg_9v_LM317
Q2
D30
Slave_device
R10
R20
R30
R16
Q2
U_Relay_16A
P2
D3
R1
R2
U_Voltage_reg_3v3_LM3171
D1
C3
D2
C1
C2
R9
P1

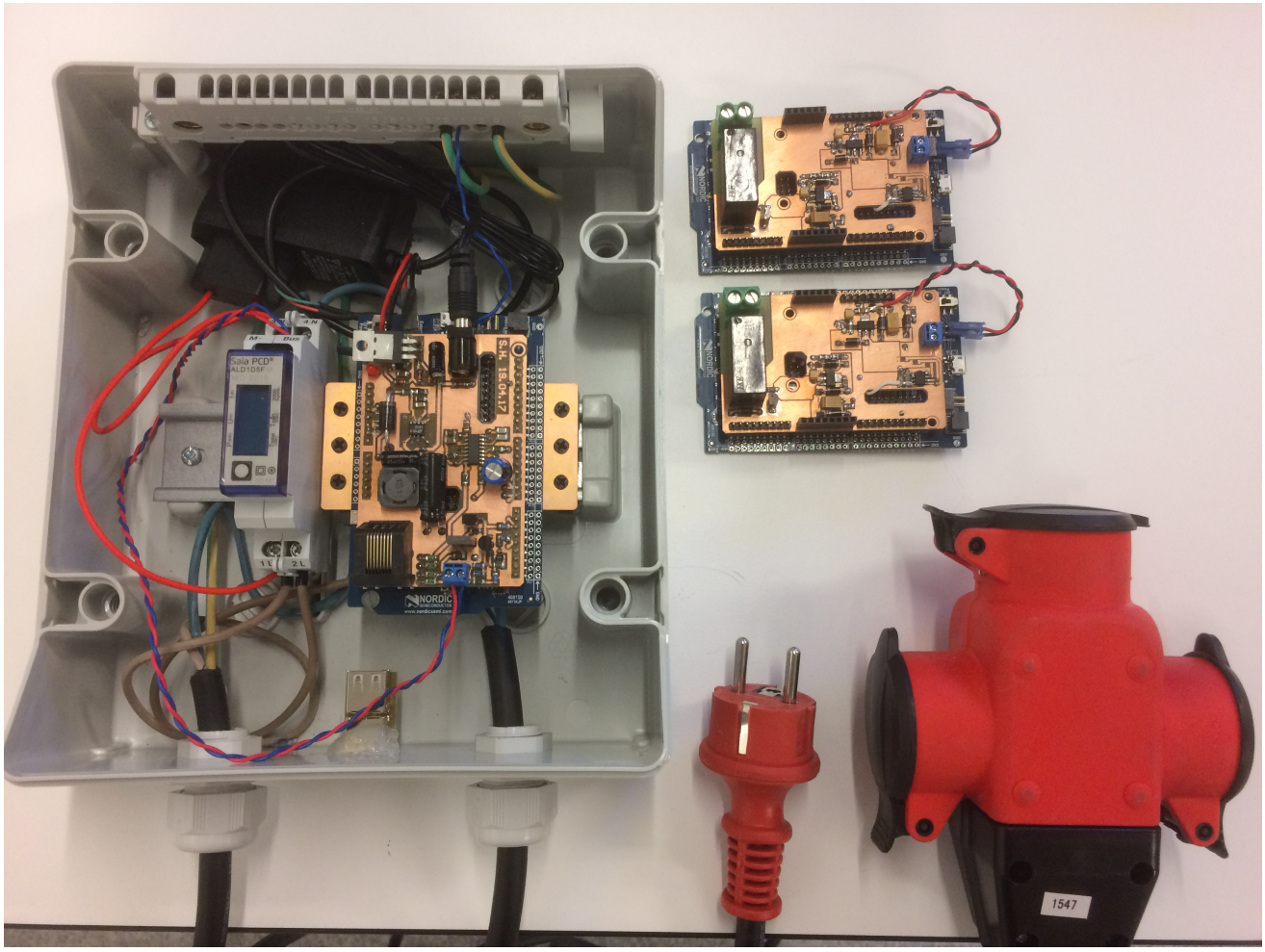

D10
C30
D20
R14
T10
D7
C13
Q4
R4
U_LM75B
R3
R5
R6
C10
U_Voltage_reg_9v_LM317
Q2
D30
R10
R20
R30
R16
Q2
U_Relay_16A
NetP1_1
NetP1_2
P2
D3
R1
R2
U_Voltage_reg_3v3_LM3171
D1
C3
D2
C1
C2
R9
P1


D10
D14
AVX
E-CASE
D20
C13
R4
R3
R5
R6
C10
C20
D30
R10
R20
R30
T1
R16
Q2
P2
D3
R1
R2
D1
D2
C1
C2
R9


AVX
E-CASE
D20
C13
R4
R3
R5
R6
D30
R10
R20
R30
T1
R16
Q2
D3
R1
R2
D1
D2
R9

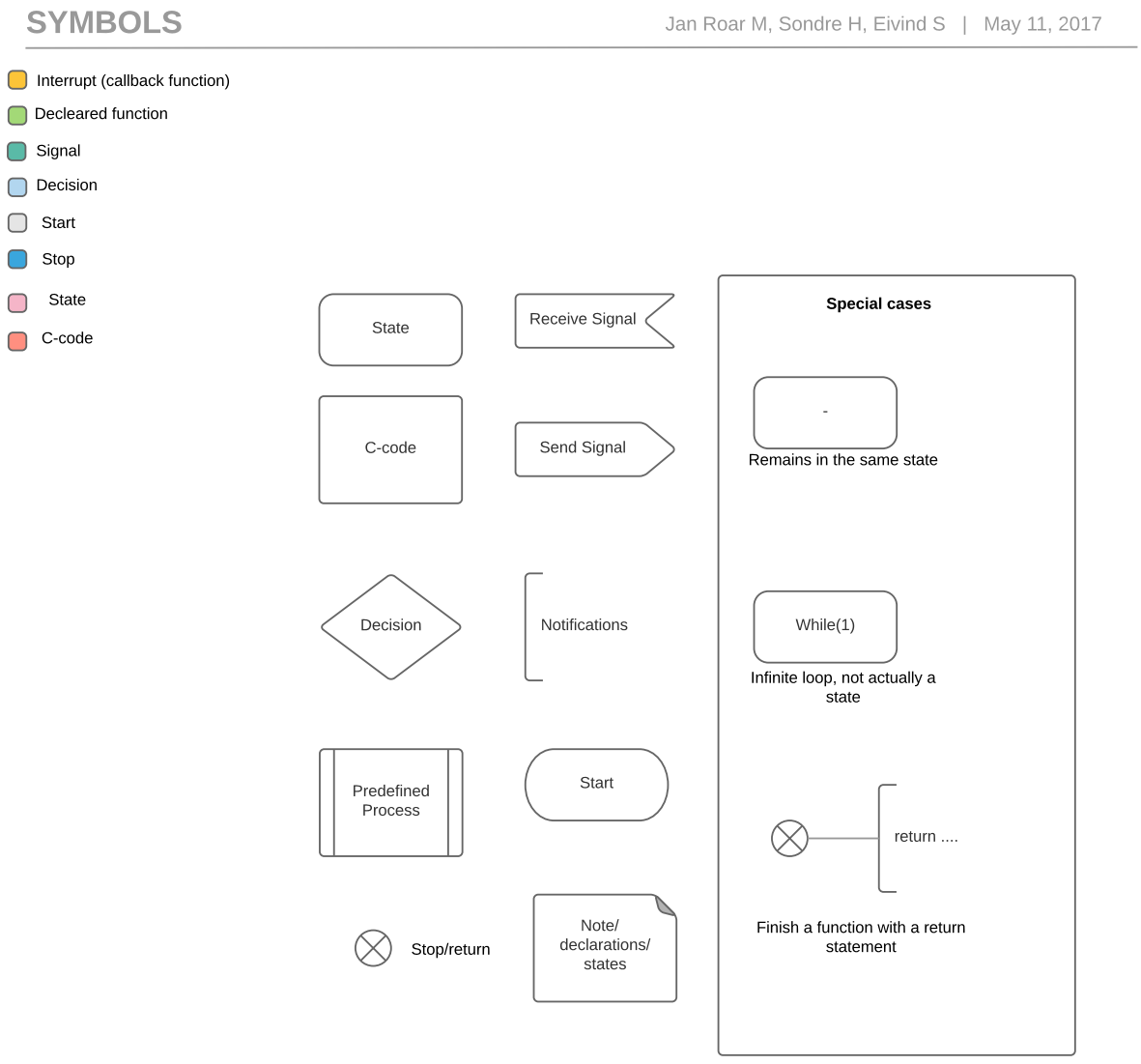


7.6.6 System hardware result

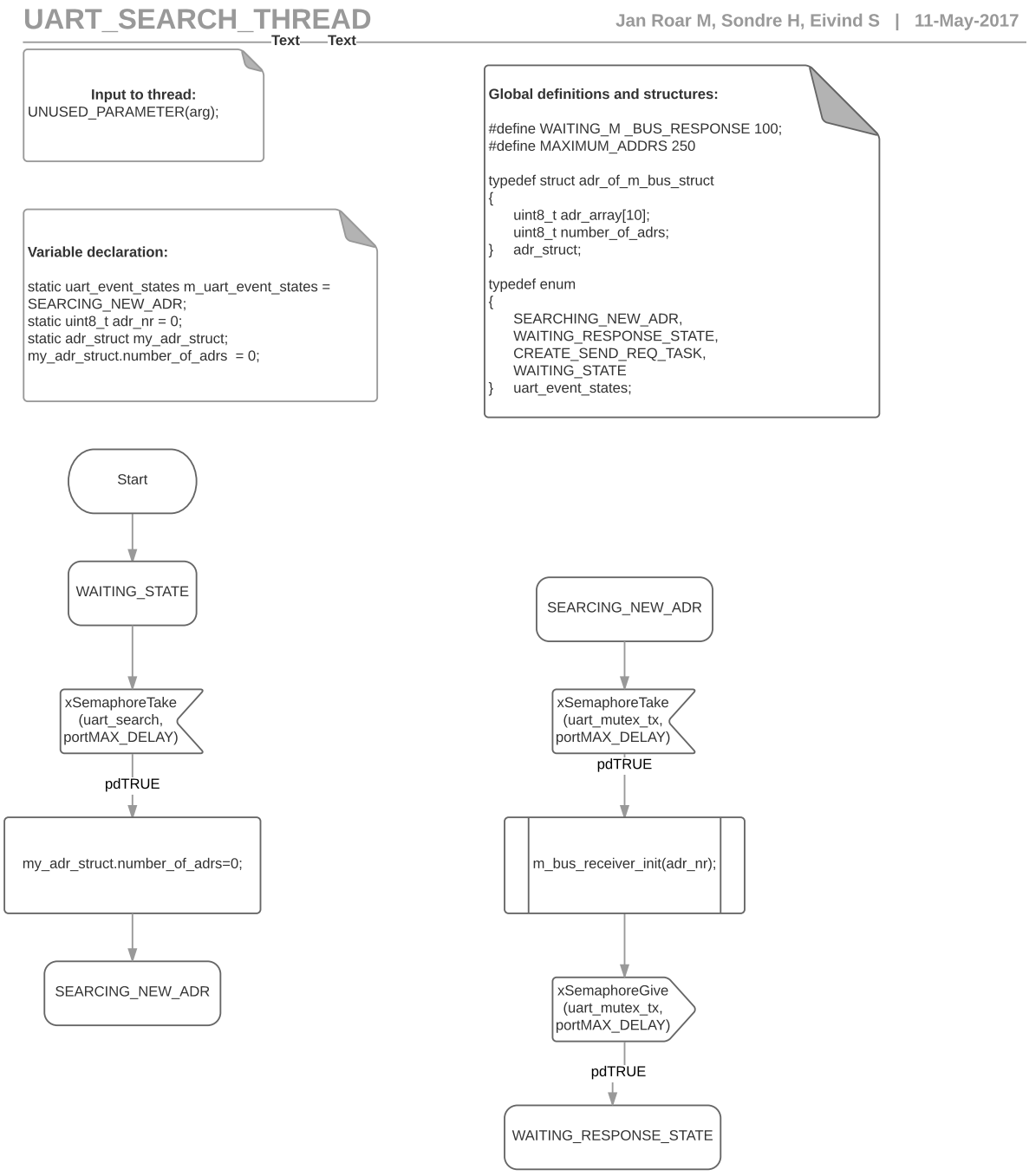


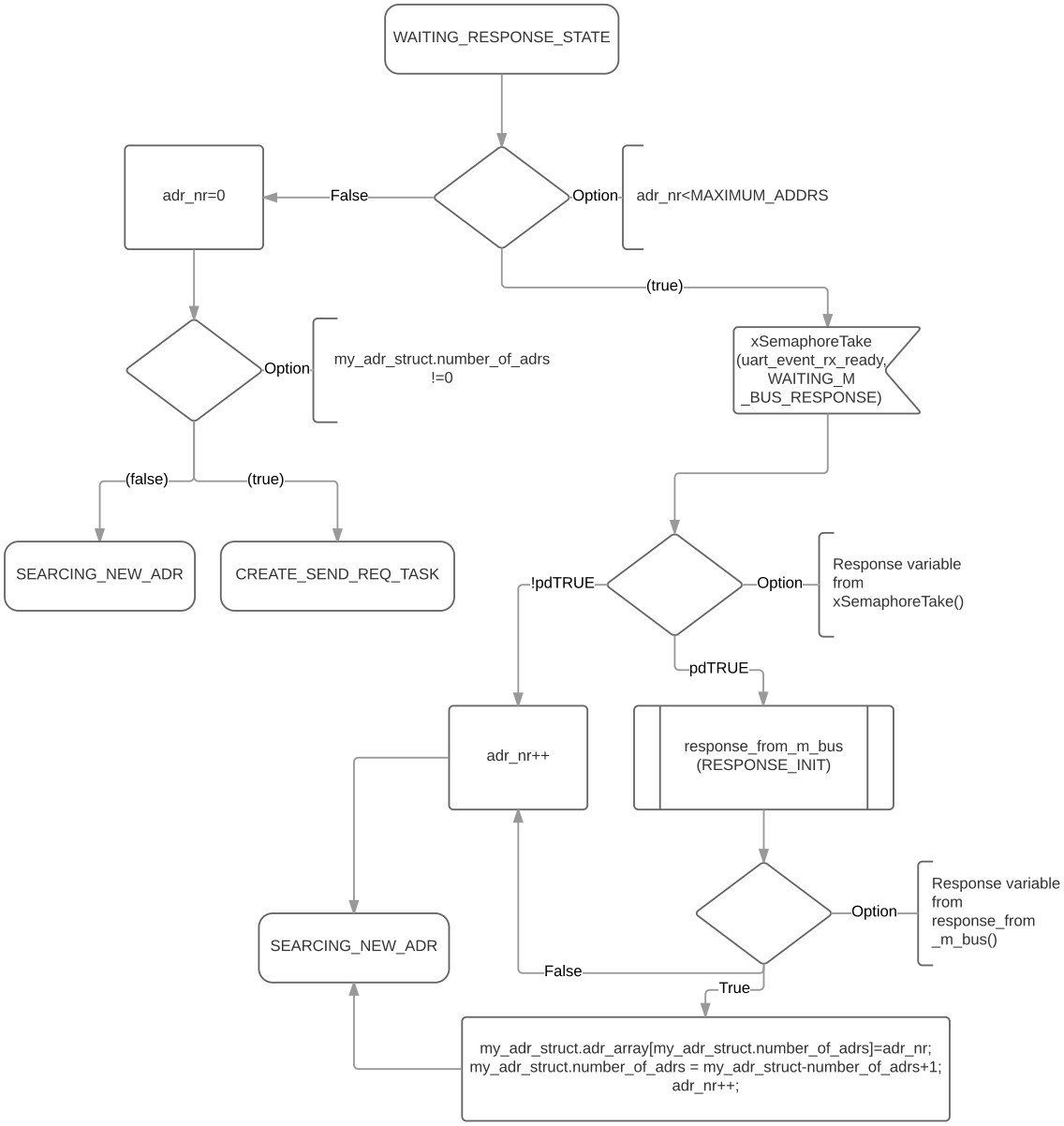
7.7 Appendix E - Software diagrams

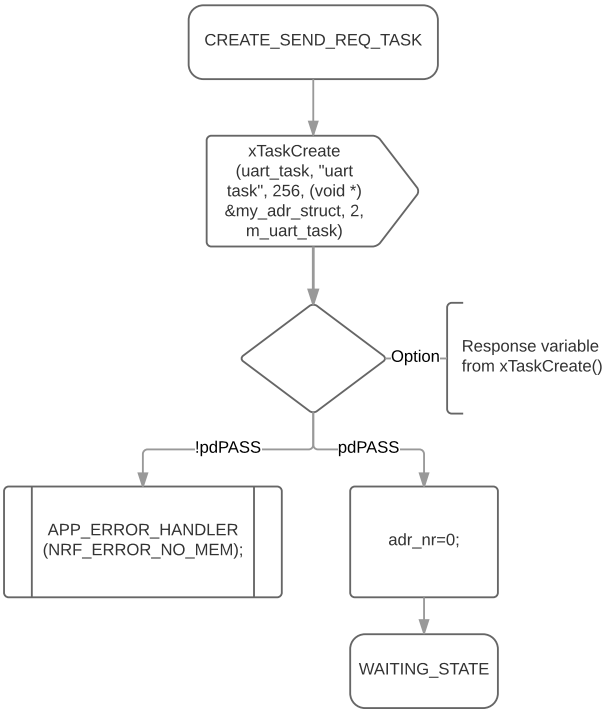
7.7.1 Symbols description



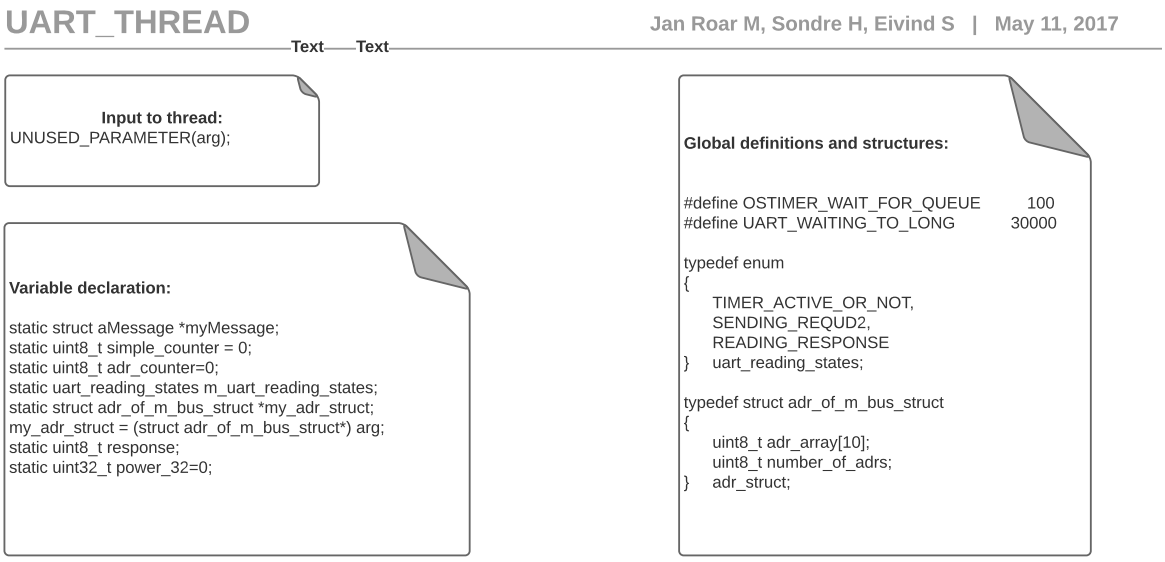
7.7.2 Thread: uart_search_thread

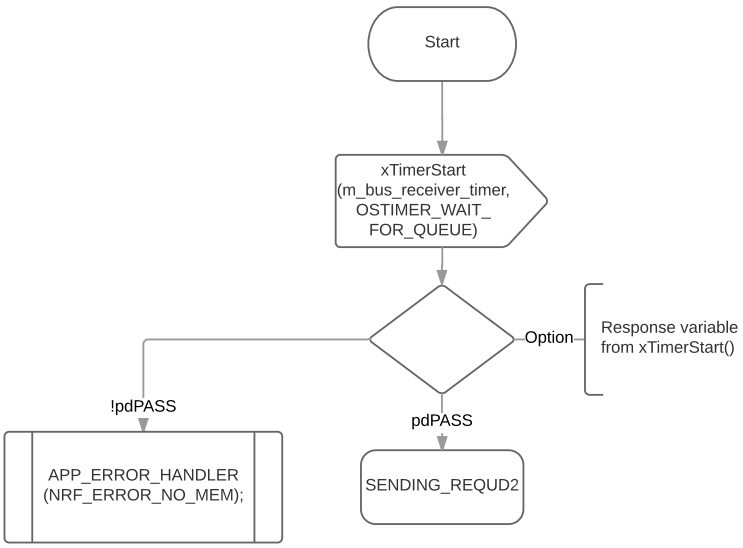


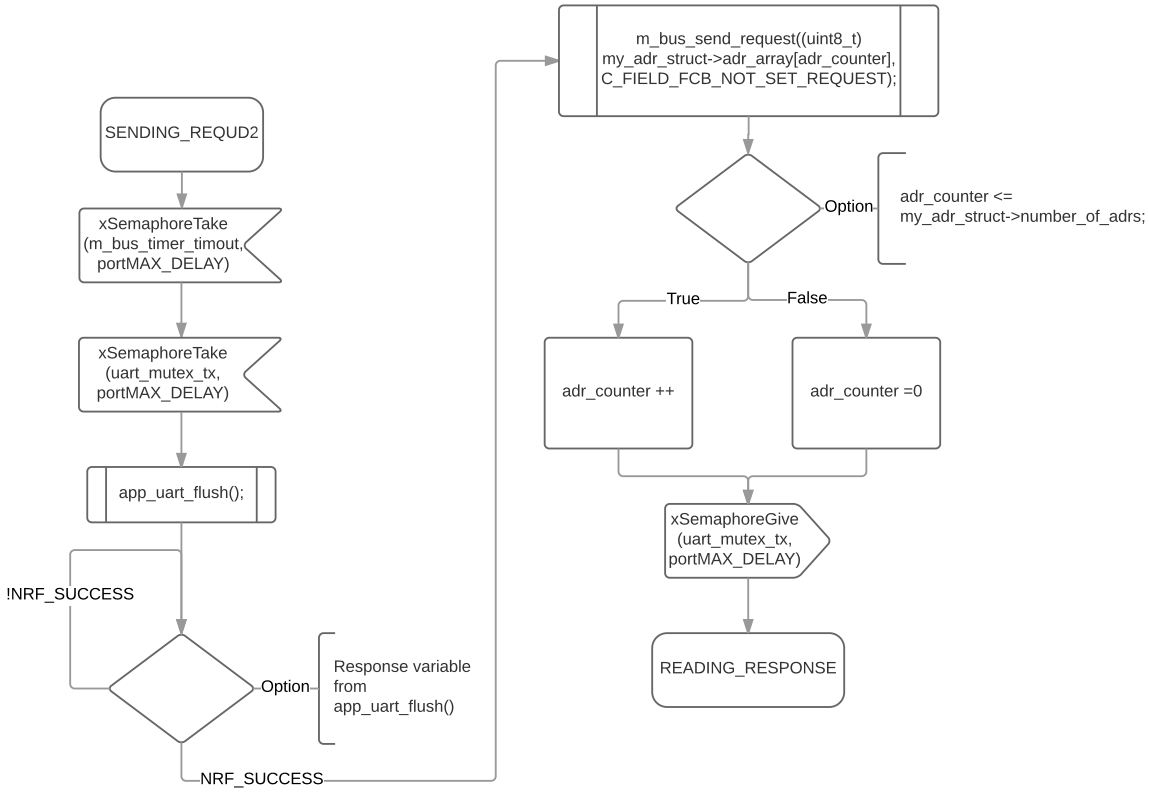

WAITING_RESPONSE_STATE
adr_nr=0
False
Option
adr_nr<MAXIMUM_ADDRS
(true)
Option
my_adr_struct.number_of_adrs
!=0
xSemaphoreTake
(uart_event_rx_ready,
WAITING_M
_BUS_RESPONSE)
(false)
(true)
SEARCING_NEW_ADR
CREATE_SEND_REQ_TASK
!pdTRUE
Option
Response variable
from
xSemaphoreTake()
pdTRUE
adr_nr++
response_from_m_bus
(RESPONSE_INIT)
Option
Response variable
from
response_from
_m_bus()
SEARCING_NEW_ADR
False
True
my_adr_struct.adr_array[my_adr_struct.number_of_adrs]=adr_nr;
my_adr_struct.number_of_adrs = my_adr_struct-number_of_adrs+1;
adr_nr++;

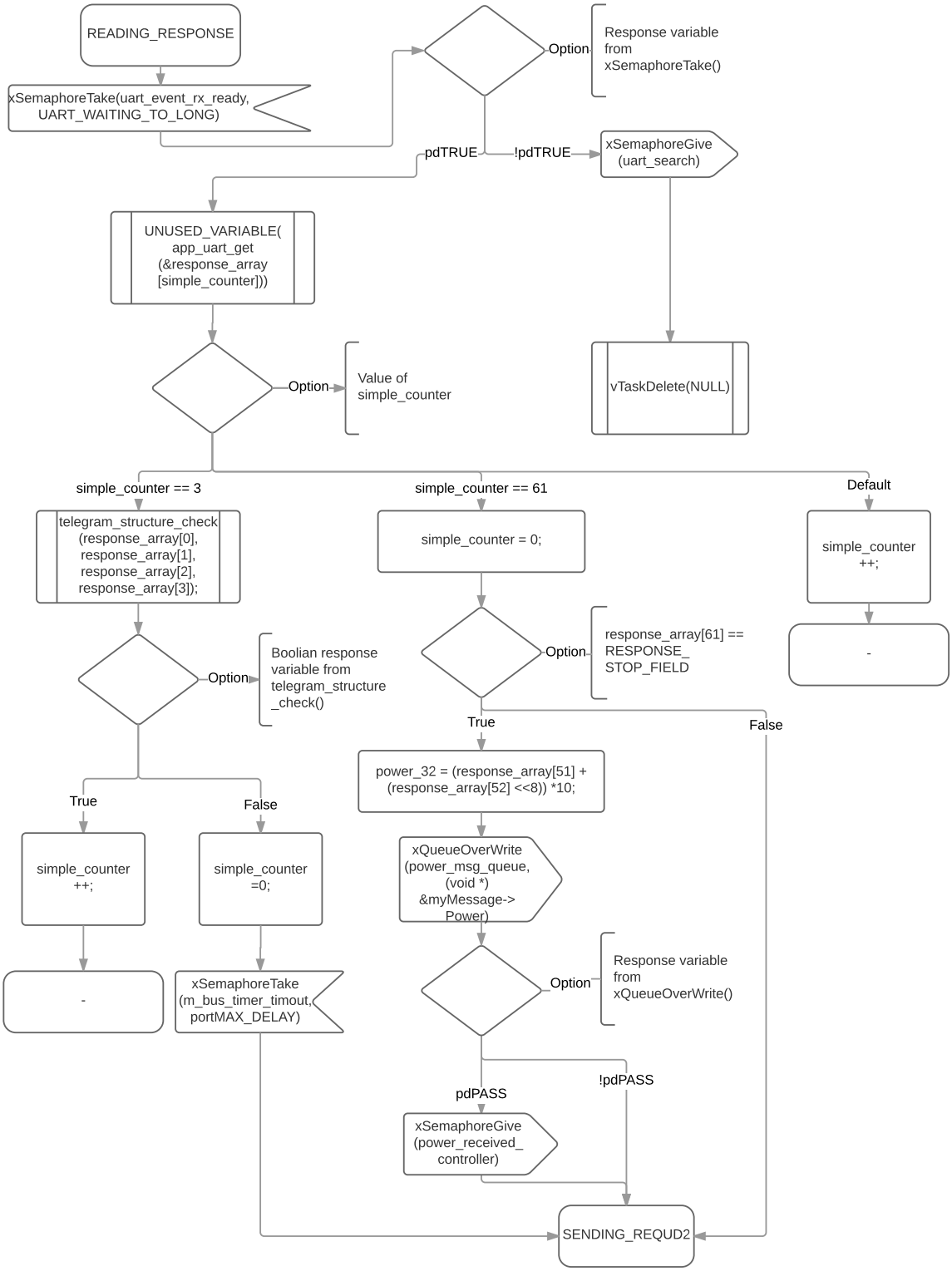

CREATE_SEND_REQ_TASK
xTaskCreate
(uart_task, "uart
task", 256, (void *)
&my_adr_struct, 2,
m_uart_task)
Option
Response variable
from xTaskCreate()
!pdPASS
pdPASS
APP_ERROR_HANDLER
(NRF_ERROR_NO_MEM);
adr_nr=0;
WAITING_STATE

7.7.3 Thread: uart_thread






SENDING_REQUD2
xSemaphoreTake (m_bus_timer_timout, portMAX_DELAY)
xSemaphoreTake (uart_mutex_tx, portMAX_DELAY)
app_uart_flush();
!NRF_SUCCESS
Option
Response variable from app_uart_flush()
NRF_SUCCESS
m_bus_send_request((uint8_t) my_adr_struct->adr_array[adr_counter], C_FIELD_FCB_NOT_SET_REQUEST);
Option
adr_counter <= my_adr_struct->number_of_adrs;
True
False
adr_counter ++
adr_counter =0
xSemaphoreGive (uart_mutex_tx, portMAX_DELAY)
READING_RESPONSE


READING_RESPONSE
xSemaphoreTake(uart_event_rx_ready, UART_WAITING_TO_LONG)
Option
Response variable from xSemaphoreTake()
pdTRUE
!pdTRUE
xSemaphoreGive (uart_search)
UNUSED_VARIABLE(app_uart_get (&response_array [simple_counter]))
Option
Value of simple_counter
vTaskDelete(NULL)
simple_counter == 3
simple_counter == 61
Default
telegram_structure_check (response_array[0], response_array[1], response_array[2], response_array[3]);
simple_counter = 0;
simple_counter ++;
Option
Boolian response variable from telegram_structure _check()
Option
response_array[61] == RESPONSE_ STOP_FIELD
-
True
False
power_32 = (response_array[51] + (response_array[52] <<8)) *10;
True
False
simple_counter ++;
simple_counter =0;
xQueueOverWrite (power_msg_queue, (void *) &myMessage-> Power)
-
xSemaphoreTake (m_bus_timer_timout, portMAX_DELAY)
Option
Response variable from xQueueOverWrite()
pdPASS
!pdPASS
xSemaphoreGive (power_received_ controller)
SENDING_REQUD2

7.7.4 Thread: controller_thread

CONTROLLER_THREAD

Jan Roar M, Sondre H, Eivind S | May 11, 2017

Text Text

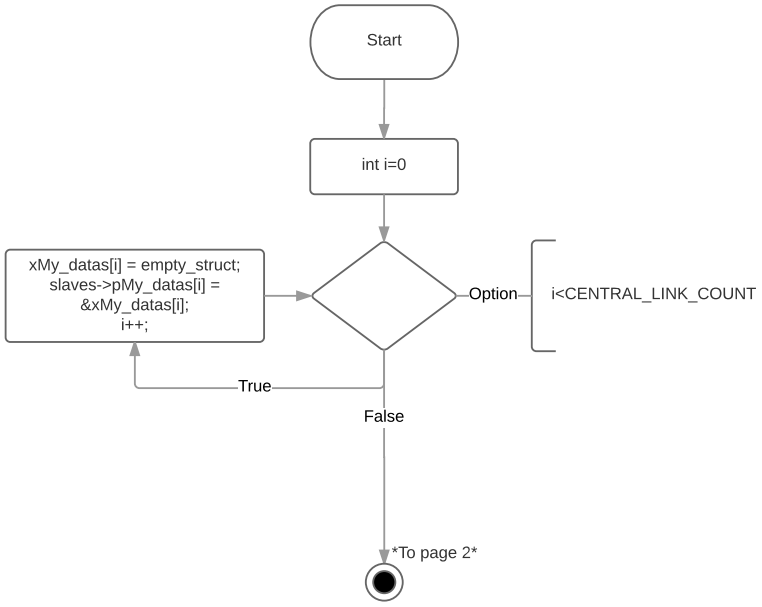
Input to thread:
UNUSED_PARAMETER(arg);

Variable declaration:

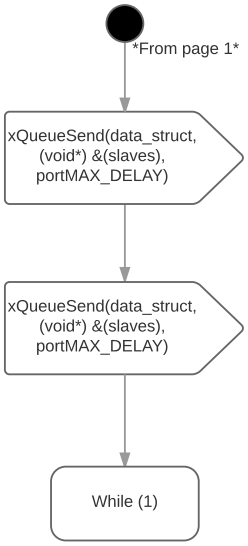
```
static uint32_t consume_diff;
static uint8_t hour;
static uint32_t power;
static uint8_t slave_nr;
static uint8_t slave_nr_clear;
static uint8_t lowest_priority_to_turn_off;

static struct My_data_pointers *slaves;
struct aMy_data xMy_datas[CENTRAL_LINK_COUNT];
slaves = &xMy_data_pointers;
static struct limits *p_limits;
p_limits = &xlimits;

p_limits->preferred_consume_limit = 2000;
p_limits->normal_max_consume_limit = 4000;
p_limits->max_consume_limit = 7000;

static struct aMy_data empty_struct;
      empty_struct.type = '0';
      empty_struct.address =0;
      empty_struct.ack =0;
      empty_struct.current_temp =0;
      empty_struct.wanted_temp =0;
      empty_struct.state =0;
      empty_struct.priority= 0;
      empty_struct.max_power =1;
```

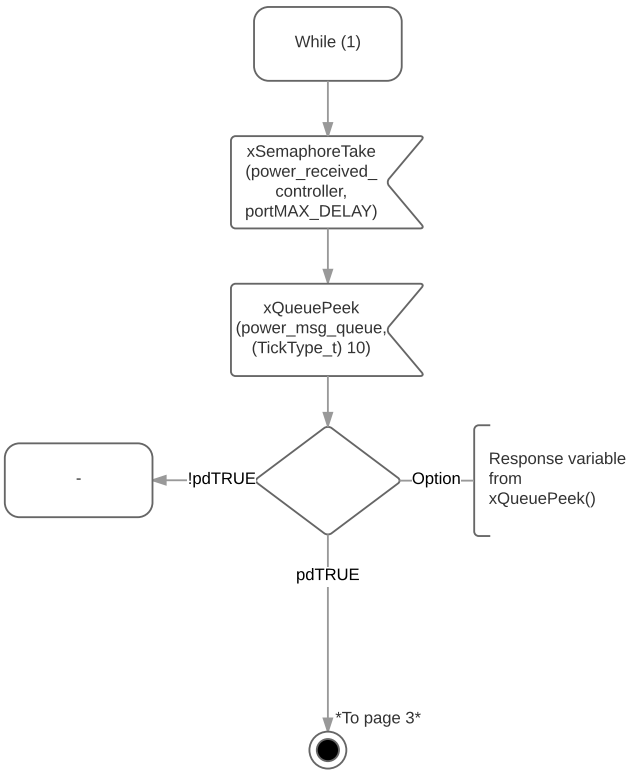
Global definitions and structures:

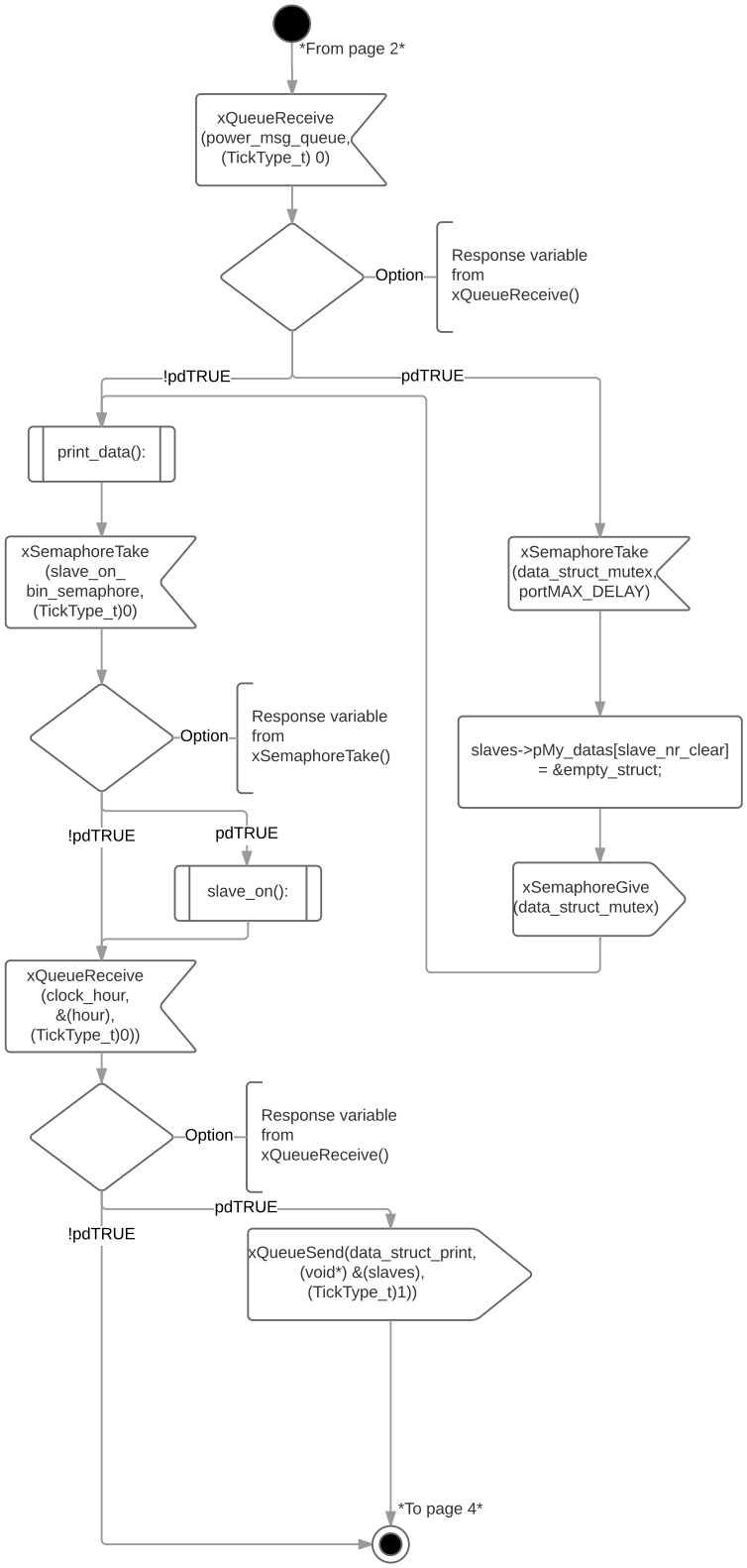
```
#define CENTRAL_LINK_COUNT     7

struct aMy_data
{
      uint8_t type;
      uint8_t address;
      uint8_t ack;
      uint8_t state;
      int8_t wanted_temp;
      int8_t current_temp;
      uint8_t priority;
      uint32_t max_power
}

struct My_data_pointers
{
      struct aMy_data
*pMy_datas[CENTRAL_LINK_COUNT];
} xMy_data_pointers;

struct limits
{
      uint32_t preferred_consume_limit;
      uint32_t normal_max_consume_limit;
      uint32_t max_consume_limit;
} xlimits;
```

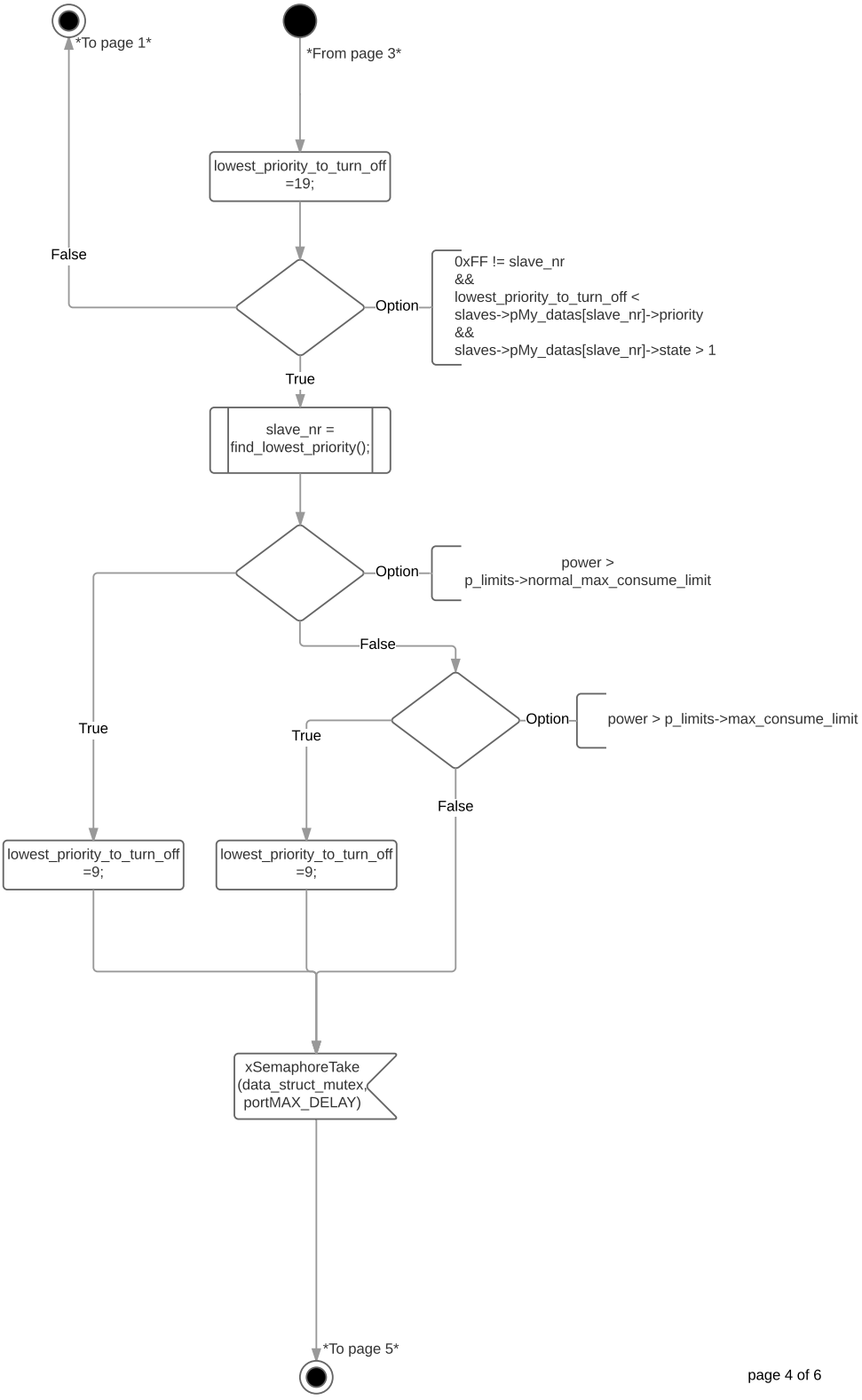


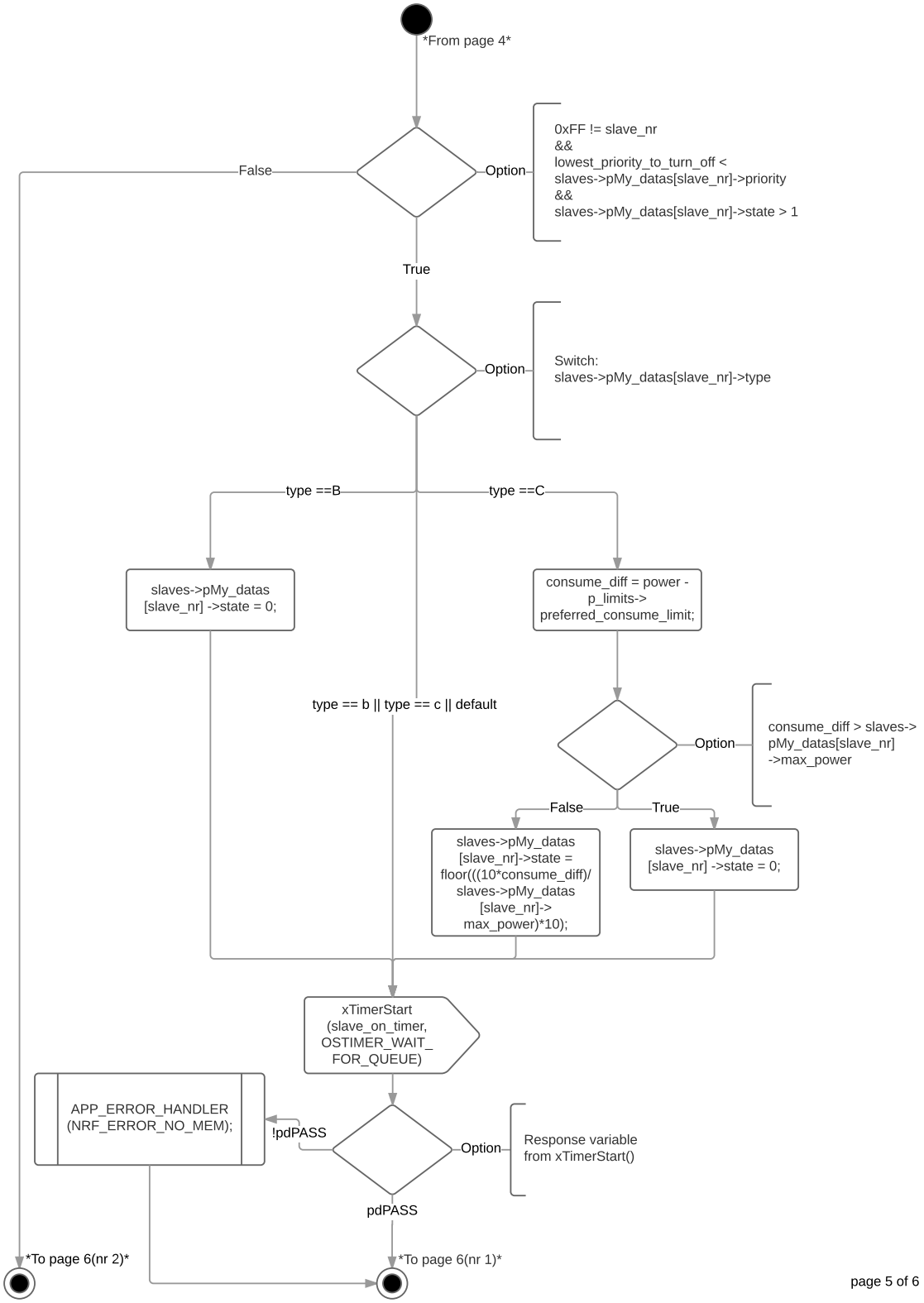
page 1 of 6

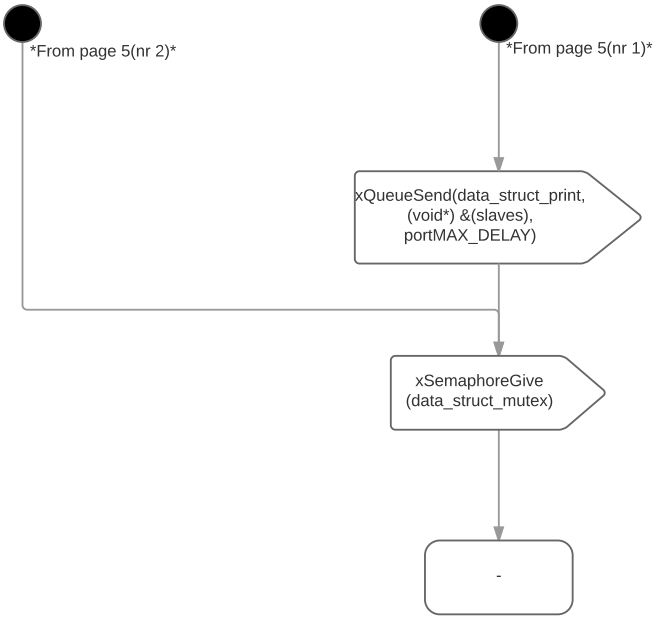

From page 1
xQueueSend(data_struct, (void*) &(slaves), portMAX_DELAY)
xQueueSend(data_struct, (void*) &(slaves), portMAX_DELAY)
While (1)

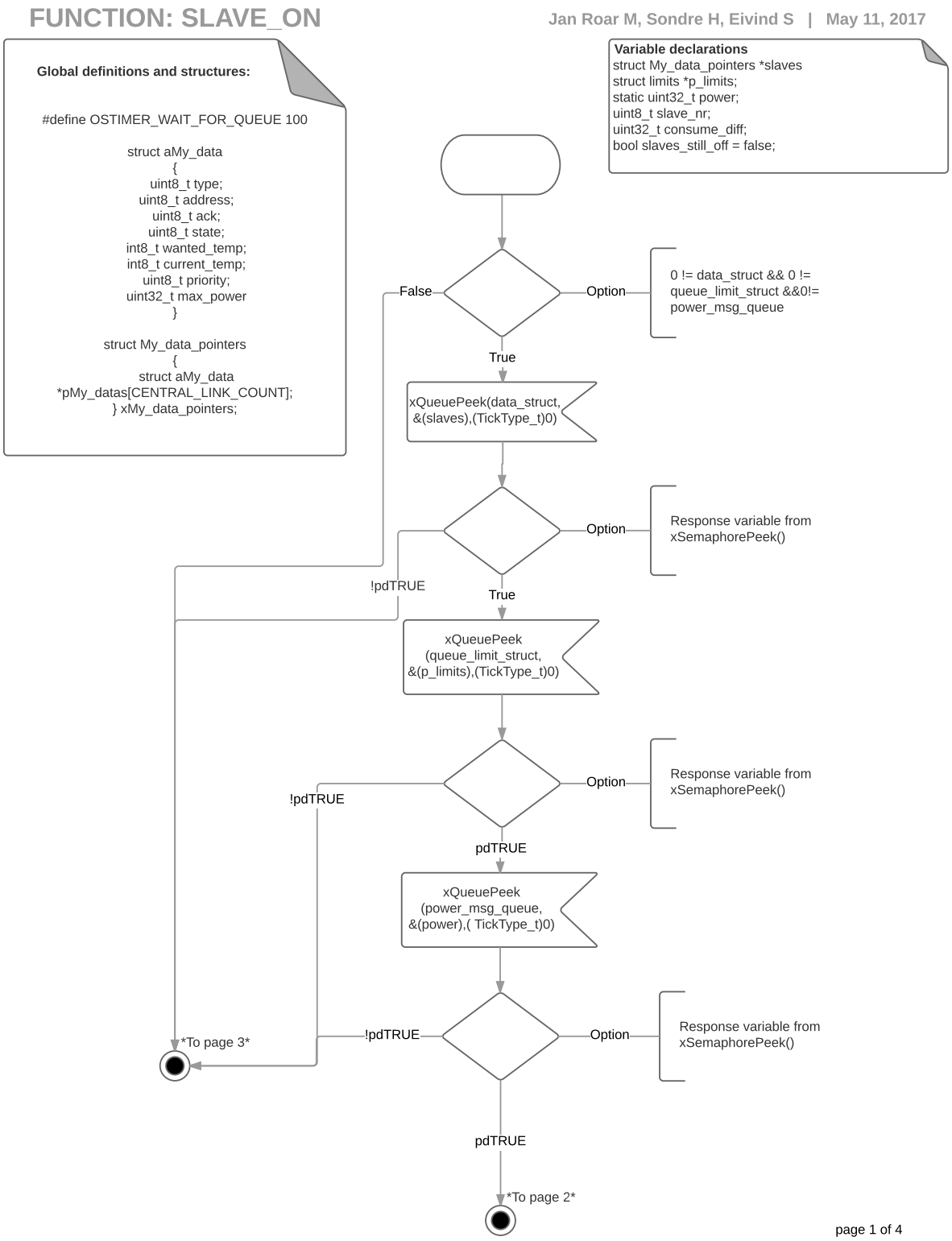

While (1)
xSemaphoreTake (power_received_ controller, portMAX_DELAY)
xQueuePeek (power_msg_queue, (TickType_t) 10)
-
!pdTRUE
Option
Response variable from xQueuePeek()
pdTRUE
To page 3

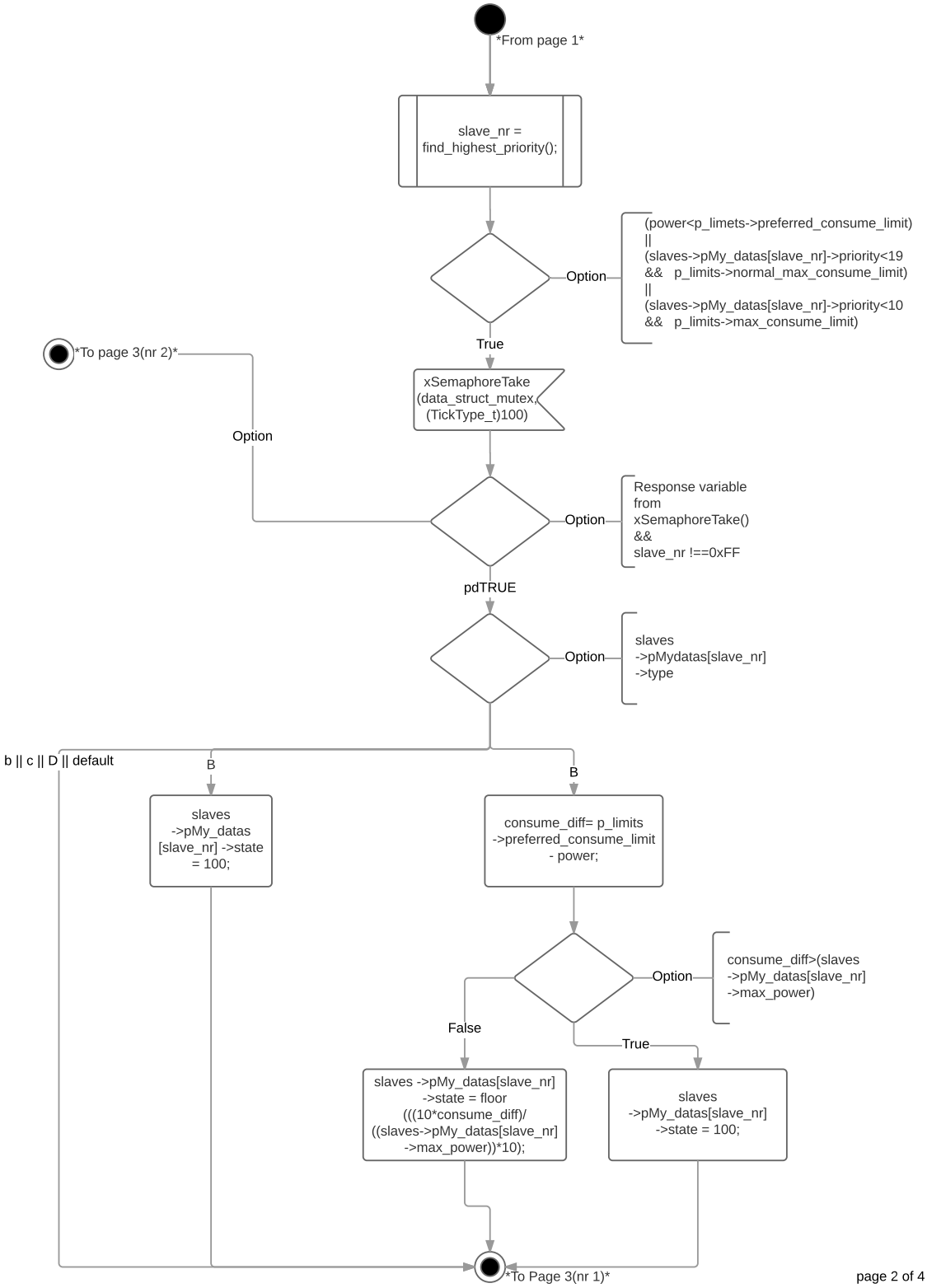

From page 2
xQueueReceive (power_msg_queue, (TickType_t) 0)
Option
Response variable from xQueueReceive()
!pdTRUE
pdTRUE
print_data():
xSemaphoreTake (slave_on_ bin_semaphore, (TickType_t)0)
xSemaphoreTake (data_struct_mutex, portMAX_DELAY)
Option
Response variable from xSemaphoreTake()
slaves->pMy_datas[slave_nr_clear] = &empty_struct;
!pdTRUE
pdTRUE
slave_on():
xSemaphoreGive (data_struct_mutex)
xQueueReceive (clock_hour, &(hour), (TickType_t)0))
Option
Response variable from xQueueReceive()
pdTRUE
!pdTRUE
xQueueSend(data_struct_print, (void*) &(slaves), (TickType_t)1))
To page 4

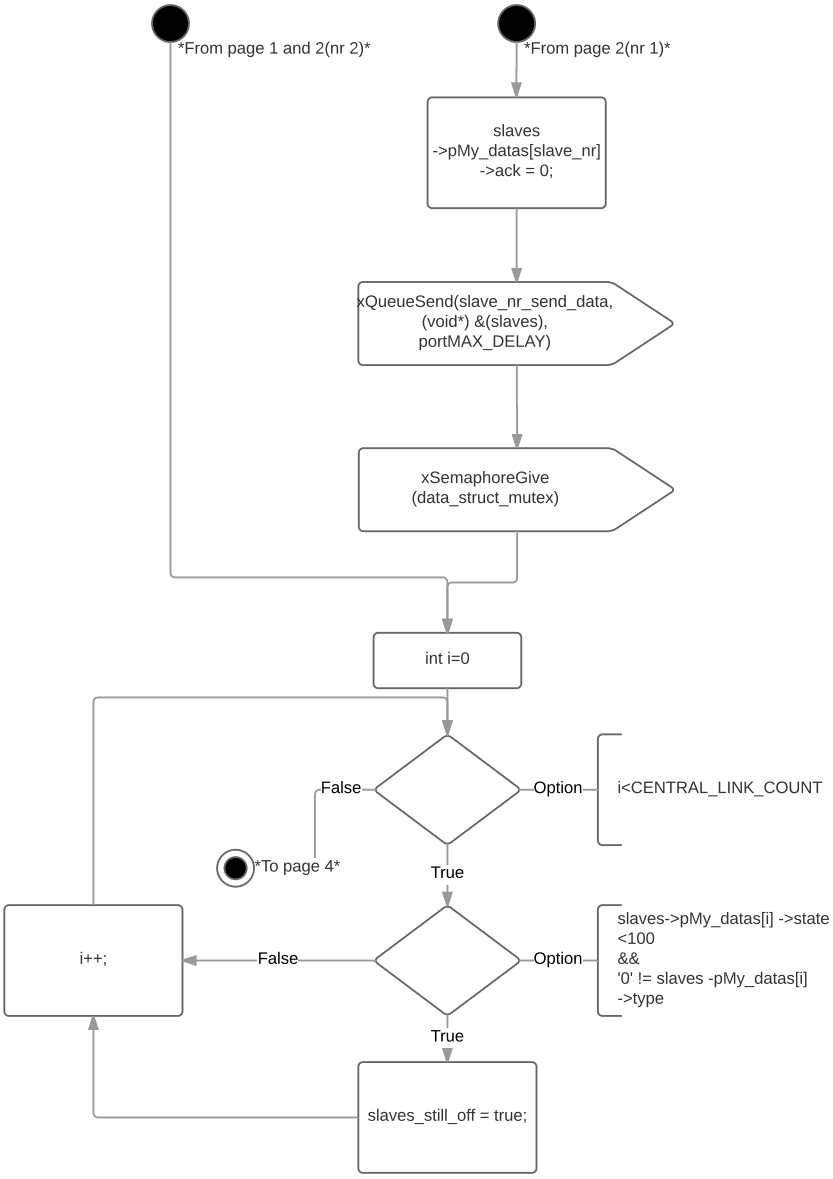

To page 1
From page 3
lowest_priority_to_turn_off =19;
False
Option
0xFF != slave_nr && lowest_priority_to_turn_off < slaves->pMy_datas[slave_nr]->priority && slaves->pMy_datas[slave_nr]->state > 1
True
slave_nr = find_lowest_priority();
Option
power > p_limits->normal_max_consume_limit
False
True
Option
power > p_limits->max_consume_limit
True
False
lowest_priority_to_turn_off =9;
lowest_priority_to_turn_off =9;
xSemaphoreTake (data_struct_mutex, portMAX_DELAY)
To page 5

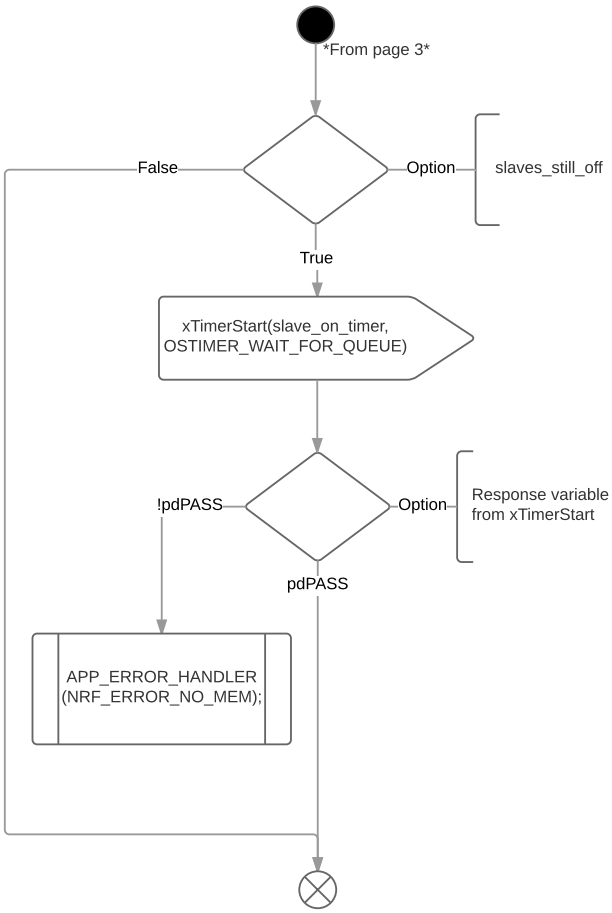

From page 4
False
Option
0xFF != slave_nr
&&
lowest_priority_to_turn_off <
slaves->pMy_datas[slave_nr]->priority
&&
slaves->pMy_datas[slave_nr]->state > 1
True
Option
Switch:
slaves->pMy_datas[slave_nr]->type
type ==B
type ==C
slaves->pMy_datas
[slave_nr] ->state = 0;
consume_diff = power -
p_limits->
preferred_consume_limit;
type == b || type == c || default
Option
consume_diff > slaves->
pMy_datas[slave_nr]
->max_power
False
True
slaves->pMy_datas
[slave_nr]->state =
floor(((10*consume_diff)/
slaves->pMy_datas
[slave_nr]->
max_power)*10);
slaves->pMy_datas
[slave_nr] ->state = 0;
xTimerStart
(slave_on_timer,
OSTIMER_WAIT_
FOR_QUEUE)
APP_ERROR_HANDLER
(NRF_ERROR_NO_MEM);
!pdPASS
Option
Response variable
from xTimerStart()
pdPASS
To page 6(nr 2)
To page 6(nr 1)


From page 5(nr 2)
From page 5(nr 1)
xQueueSend(data_struct_print,
(void*) &(slaves),
portMAX_DELAY)
xSemaphoreGive
(data_struct_mutex)
-


FUNCTION: SLAVE_ON
Jan Roar M, Sondre H, Eivind S | May 11, 2017
Global definitions and structures:
#define OSTIMER_WAIT_FOR_QUEUE 100
struct aMy_data
{
uint8_t type;
uint8_t address;
uint8_t ack;
uint8_t state;
int8_t wanted_temp;
int8_t current_temp;
uint8_t priority;
uint32_t max_power
}
struct My_data_pointers
{
struct aMy_data
*pMy_datas[CENTRAL_LINK_COUNT];
} xMy_data_pointers;
Variable declarations
struct My_data_pointers *slaves
struct limits *p_limits;
static uint32_t power;
uint8_t slave_nr;
uint32_t consume_diff;
bool slaves_still_off = false;
False
Option
0 != data_struct && 0 != queue_limit_struct &&0!= power_msg_queue
True
xQueuePeek(data_struct, &(slaves),(TickType_t)0)
Option
Response variable from xSemaphorePeek()
!pdTRUE
True
xQueuePeek (queue_limit_struct, &(p_limits),(TickType_t)0)
Option
Response variable from xSemaphorePeek()
!pdTRUE
pdTRUE
xQueuePeek (power_msg_queue, &(power),(TickType_t)0)
To page 3
!pdTRUE
Option
Response variable from xSemaphorePeek()
pdTRUE
To page 2

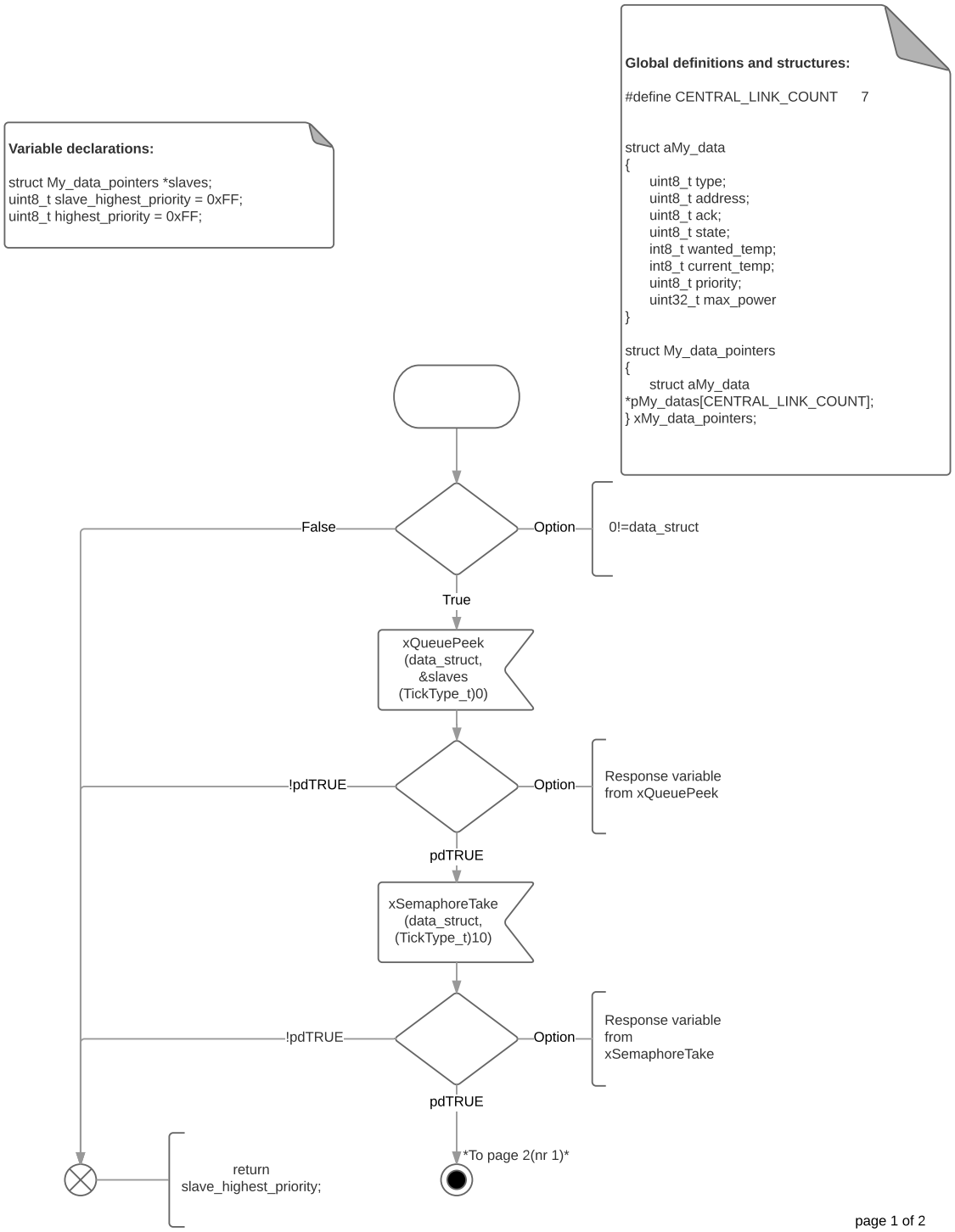

From page 1
slave_nr =
find_highest_priority();
Option
(power<p_limets->preferred_consume_limit)
||
(slaves->pMy_datas[slave_nr]->priority<19
&& p_limits->normal_max_consume_limit)
||
(slaves->pMy_datas[slave_nr]->priority<10
&& p_limits->max_consume_limit)
True
To page 3(nr 2)
xSemaphoreTake
(data_struct_mutex,
(TickType_t)100)
Option
Option
Response variable
from
xSemaphoreTake()
&&
slave_nr !==0xFF
pdTRUE
Option
slaves
->pMydatas[slave_nr]
->type
b || c || D || default
B
B
slaves
->pMy_datas
[slave_nr] ->state
= 100;
consume_diff= p_limits
->preferred_consume_limit
- power;
Option
consume_diff>(slaves
->pMy_datas[slave_nr]
->max_power)
False
True
slaves ->pMy_datas[slave_nr]
->state = floor
(((10*consume_diff)/
((slaves->pMy_datas[slave_nr]
->max_power))*10);
slaves
->pMy_datas[slave_nr]
->state = 100;
To Page 3(nr 1)

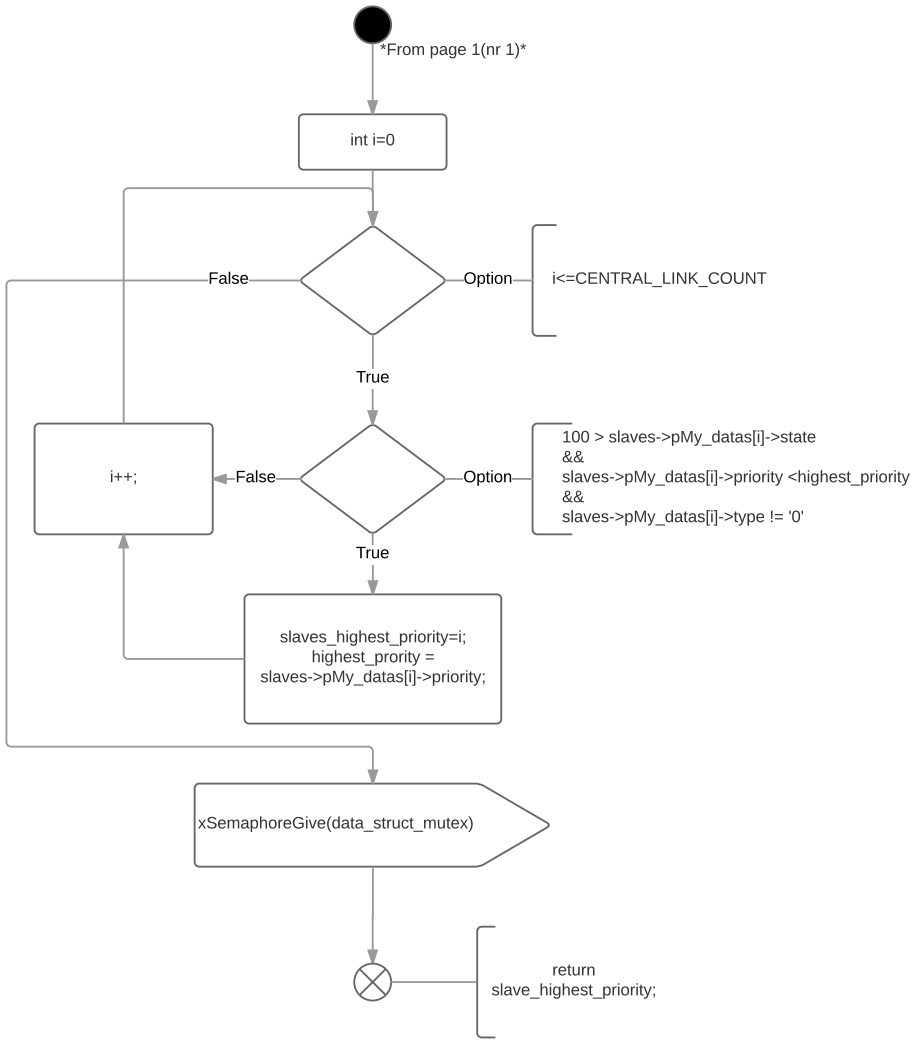

From page 1 and 2(nr 2)
From page 2(nr 1)
slaves
->pMy_datas[slave_nr]
->ack = 0;
xQueueSend(slave_nr_send_data,
(void*) &(slaves),
portMAX_DELAY)
xSemaphoreGive
(data_struct_mutex)
int i=0
False
Option
i<CENTRAL_LINK_COUNT
To page 4
True
i++;
False
Option
slaves->pMy_datas[i] ->state
<100
&&
'0' != slaves -pMy_datas[i]
->type
True
slaves_still_off = true;


From page 3
False
Option
slaves_still_off
True
xTimerStart(slave_on_timer,
OSTIMER_WAIT_FOR_QUEUE)
!pdPASS
Option
Response variable
from xTimerStart
pdPASS
APP_ERROR_HANDLER
(NRF_ERROR_NO_MEM);

FUNCTION: FIND_HIGHEST_PRIORITY

Jan Roar M, Sondre H, Eivind S | May 11, 2017




From page 1(nr 1)
int i=0
False
Option
i<=CENTRAL_LINK_COUNT
True
i++;
False
Option
100 > slaves->pMy_datas[i]->state
&&
slaves->pMy_datas[i]->priority <highest_priority
&&
slaves->pMy_datas[i]->type != '0'
True
slaves_highest_priority=i;
highest_prority =
slaves->pMy_datas[i]->priority;
xSemaphoreGive(data_struct_mutex)
return
slave_highest_priority;

FUNCTION: FIND_LOWEST_PRIORITY

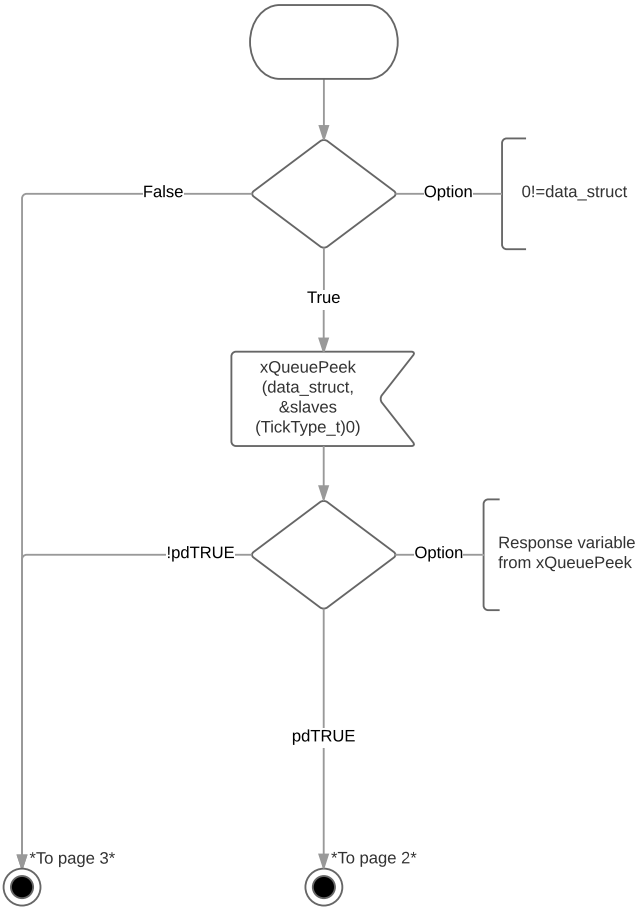
Jan Roar M, Sondre H, Eivind S | May 11, 2017

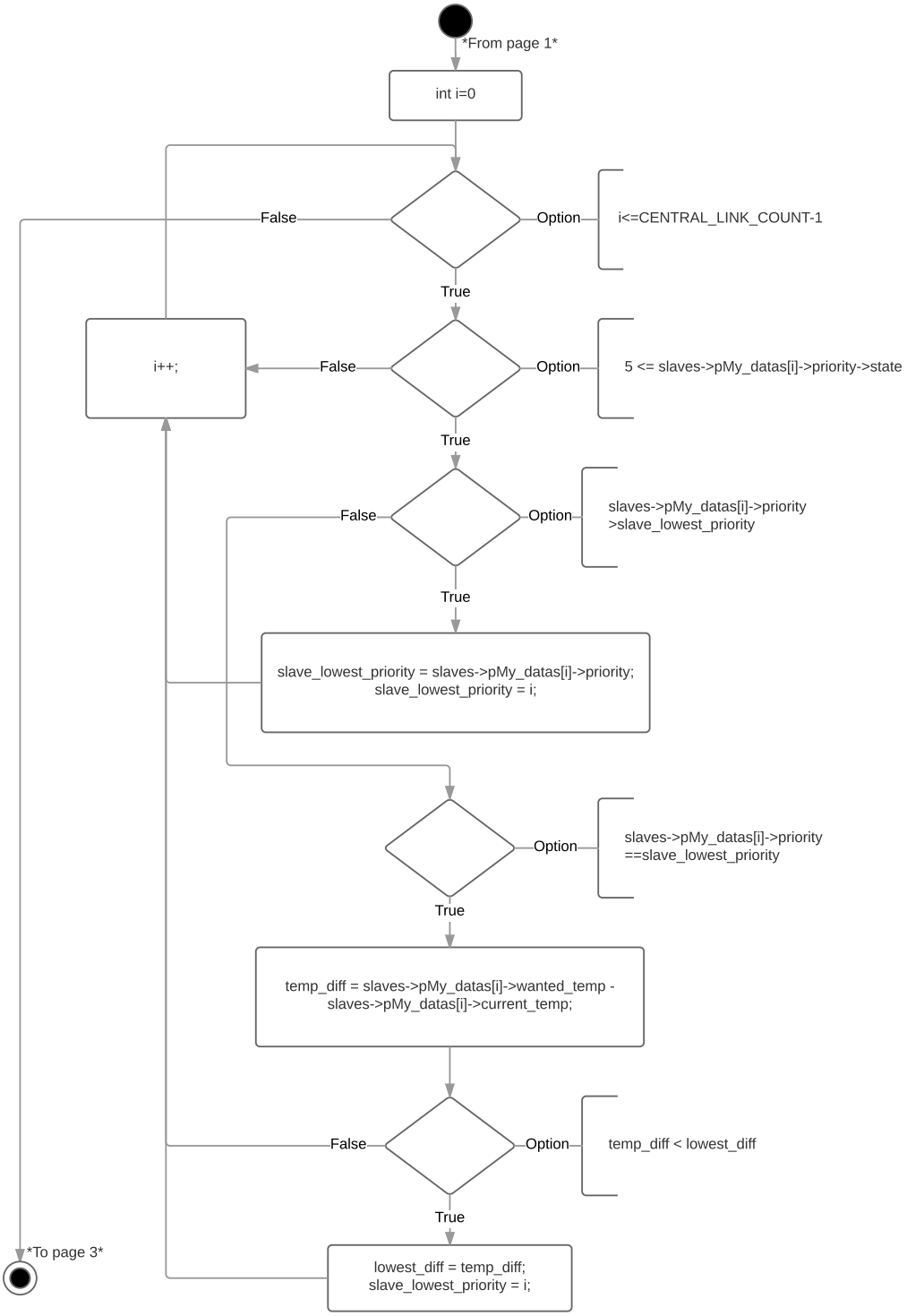
Global definitions and structures:

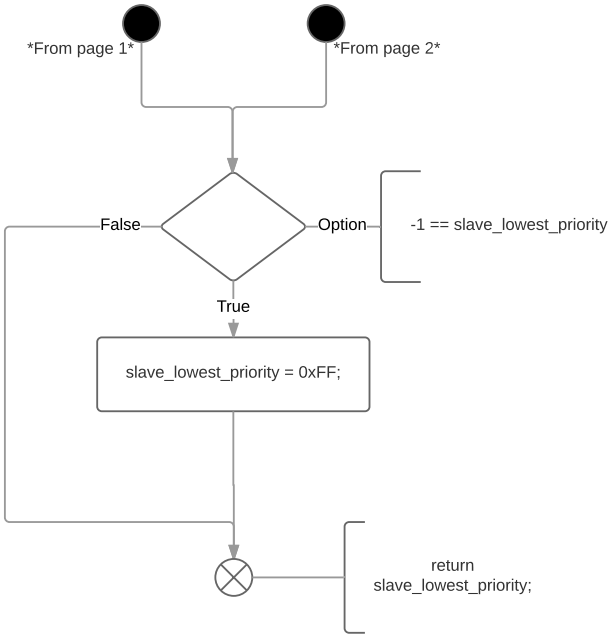
```
#define CENTRAL_LINK_COUNT     7


struct aMy_data
{
      uint8_t type;
      uint8_t address;
      uint8_t ack;
      uint8_t state;
      int8_t wanted_temp;
      int8_t current_temp;
      uint8_t priority;
      uint32_t max_power
}

struct My_data_pointers
{
      struct aMy_data
*pMy_datas[CENTRAL_LINK_COUNT];
} xMy_data_pointers;
```

Variable declarations:

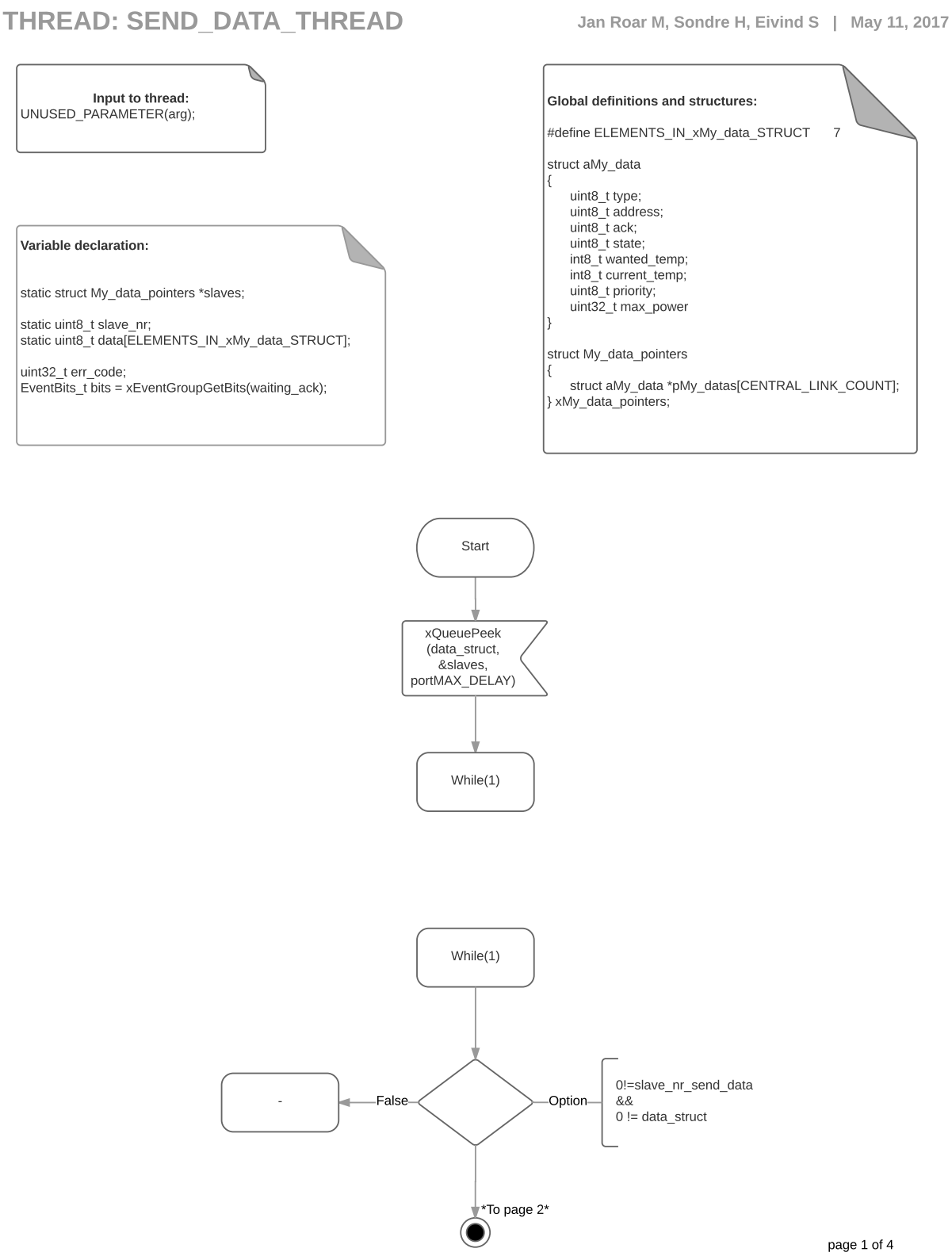
```
struct My_data_pointers *slaves;
uint8_t slave_lowest_priority = -1;
uint8_t lowest_diff = 100;
uint8_t temp_diff;
```

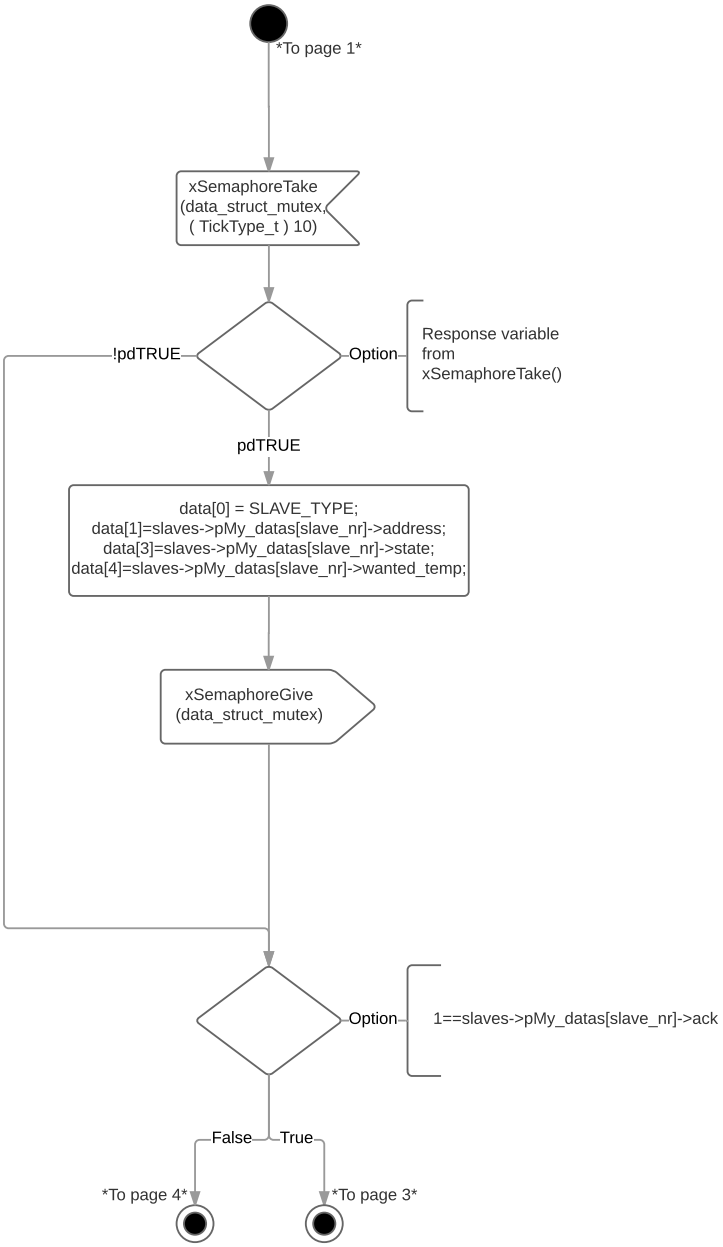


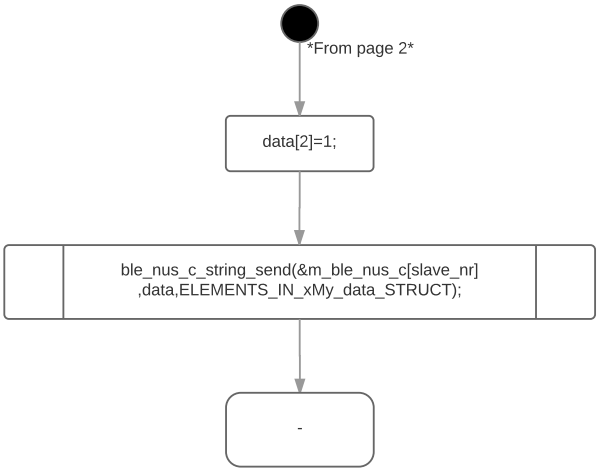

From page 1
int i=0
False
Option
i<=CENTRAL_LINK_COUNT-1
True
i++;
False
Option
5 <= slaves->pMy_datas[i]->priority->state
True
False
Option
slaves->pMy_datas[i]->priority
>slave_lowest_priority
True
slave_lowest_priority = slaves->pMy_datas[i]->priority;
slave_lowest_priority = i;
Option
slaves->pMy_datas[i]->priority
==slave_lowest_priority
True
temp_diff = slaves->pMy_datas[i]->wanted_temp -
slaves->pMy_datas[i]->current_temp;
False
Option
temp_diff < lowest_diff
True
lowest_diff = temp_diff;
slave_lowest_priority = i;
To page 3

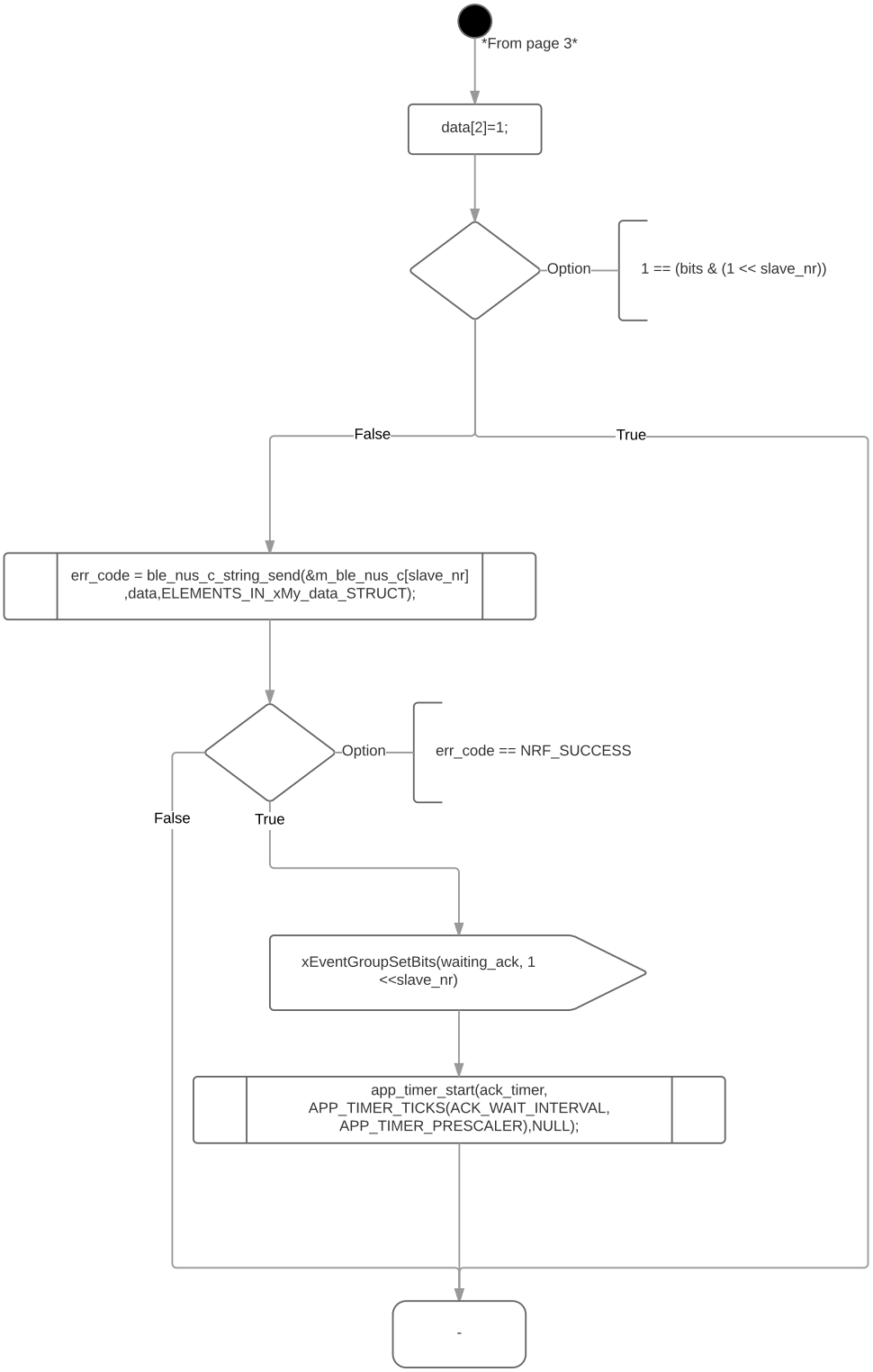

From page 1
From page 2
False
Option
-1 == slave_lowest_priority
True
slave_lowest_priority = 0xFF;
return
slave_lowest_priority;

7.7.5 Thread: send_data_thread

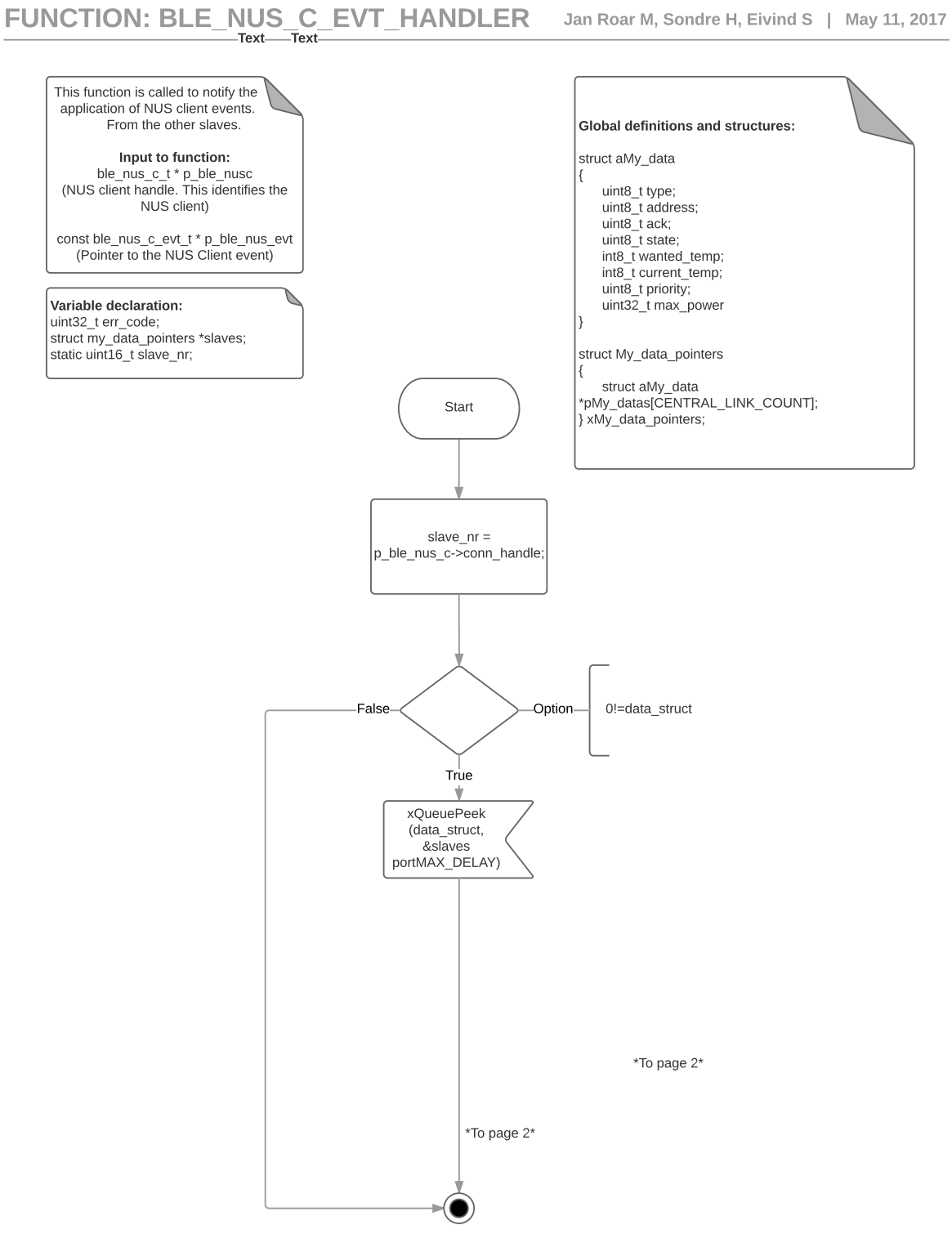


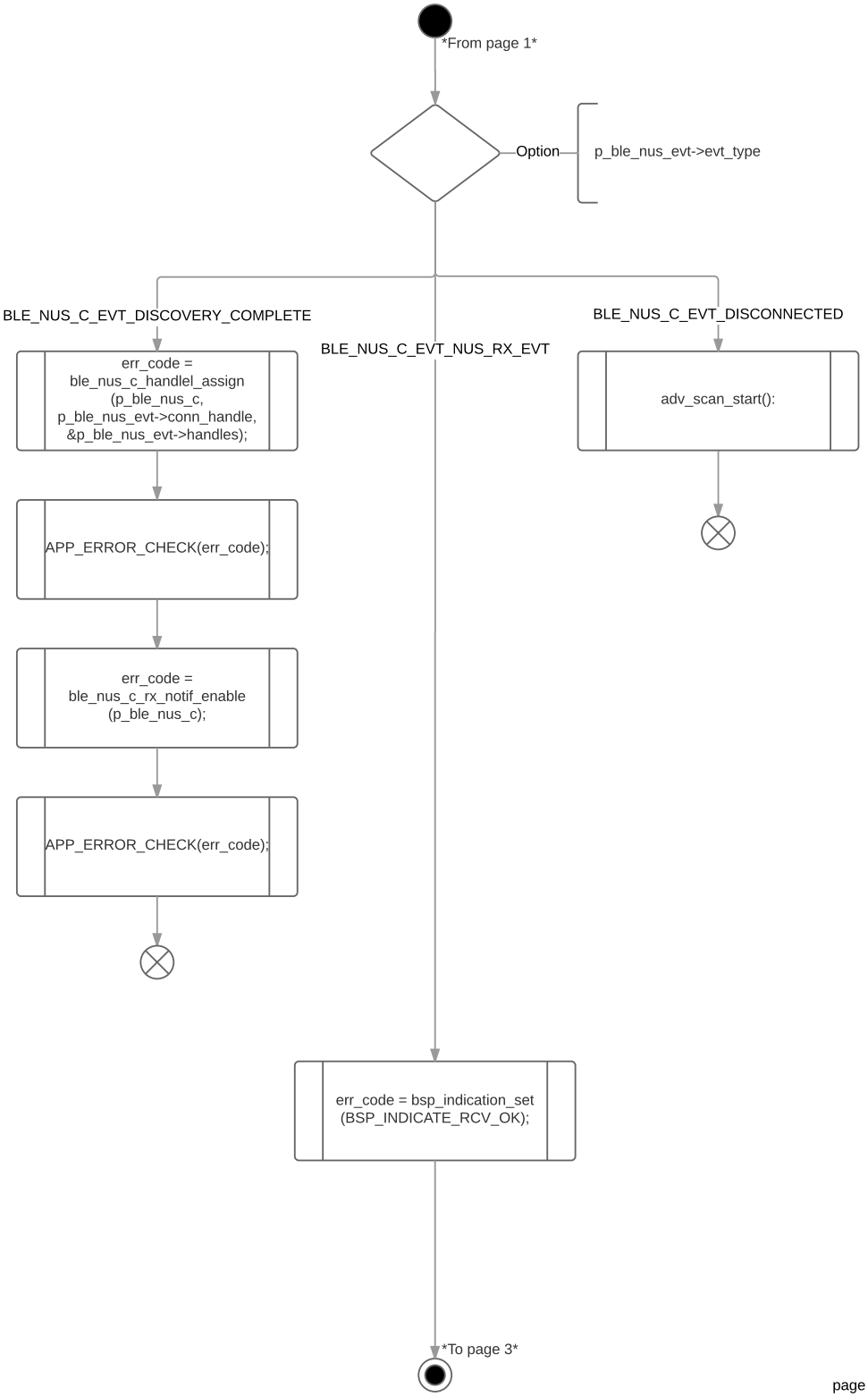

To page 1
xSemaphoreTake
(data_struct_mutex,
(TickType_t) 10)
!pdTRUE
Option
Response variable
from
xSemaphoreTake()
pdTRUE
data[0] = SLAVE_TYPE;
data[1]=slaves->pMy_datas[slave_nr]->address;
data[3]=slaves->pMy_datas[slave_nr]->state;
data[4]=slaves->pMy_datas[slave_nr]->wanted_temp;
xSemaphoreGive
(data_struct_mutex)
Option
1==slaves->pMy_datas[slave_nr]->ack
False
True
To page 4
To page 3

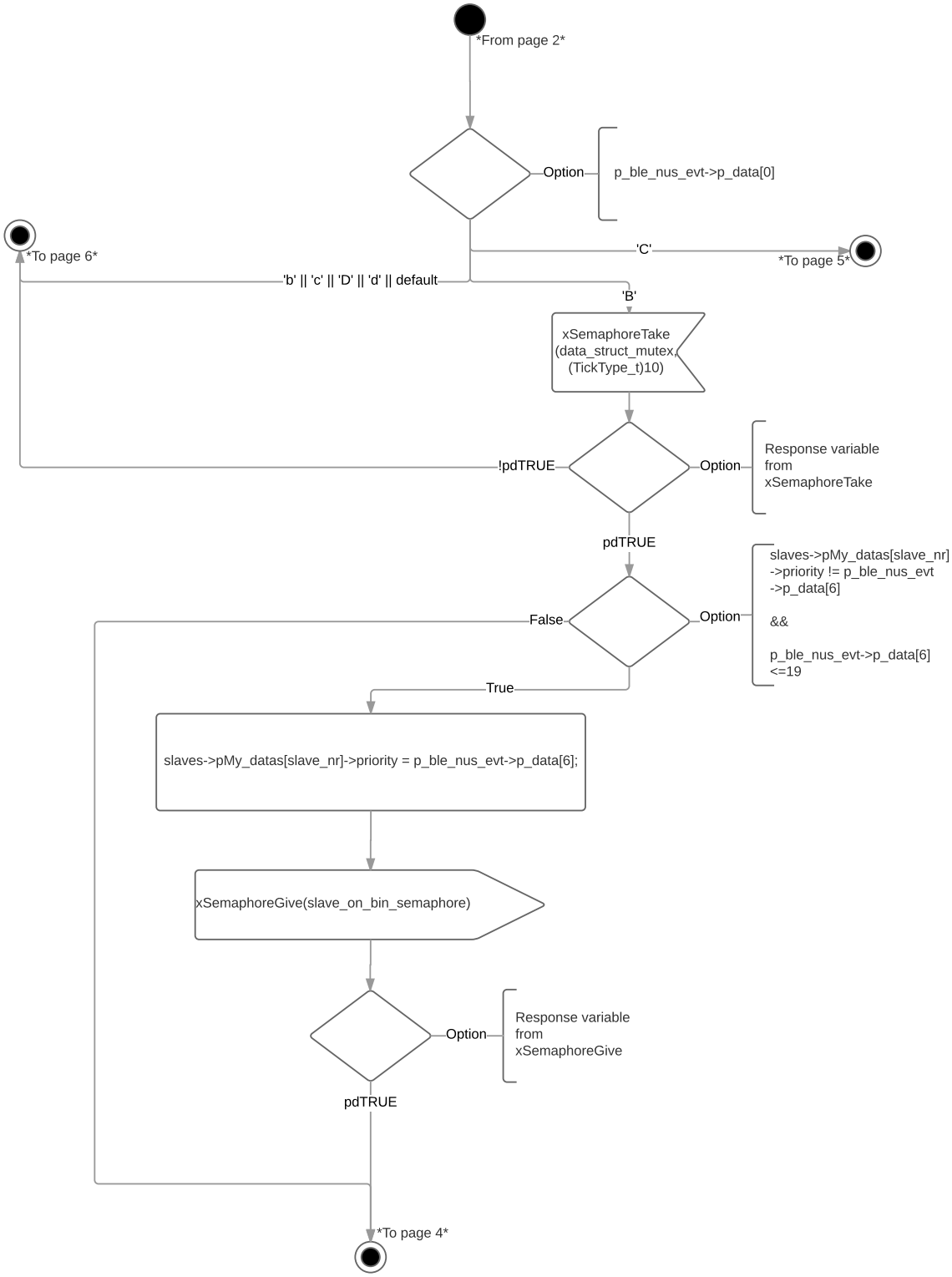

From page 2
data[2]=1;
ble_nus_c_string_send(&m_ble_nus_c[slave_nr]
,data,ELEMENTS_IN_xMy_data_STRUCT);
-

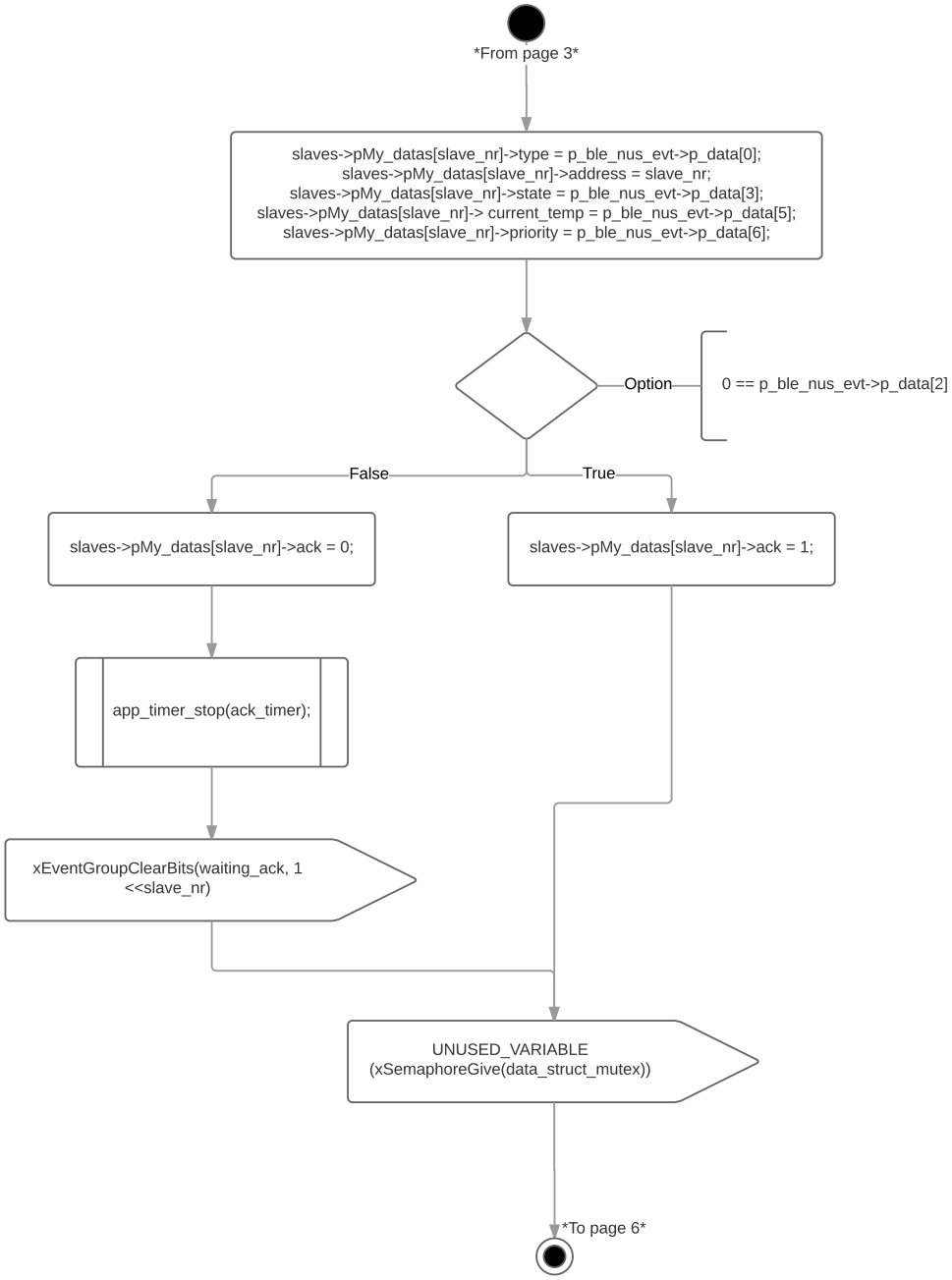

From page 3
data[2]=1;
Option
1 == (bits & (1 << slave_nr))
False
True
err_code = ble_nus_c_string_send(&m_ble_nus_c[slave_nr]
,data,ELEMENTS_IN_xMy_data_STRUCT);
Option
err_code == NRF_SUCCESS
False
True
xEventGroupSetBits(waiting_ack, 1
<<slave_nr)
app_timer_start(ack_timer,
APP_TIMER_TICKS(ACK_WAIT_INTERVAL,
APP_TIMER_PRESCALER),NULL);
-

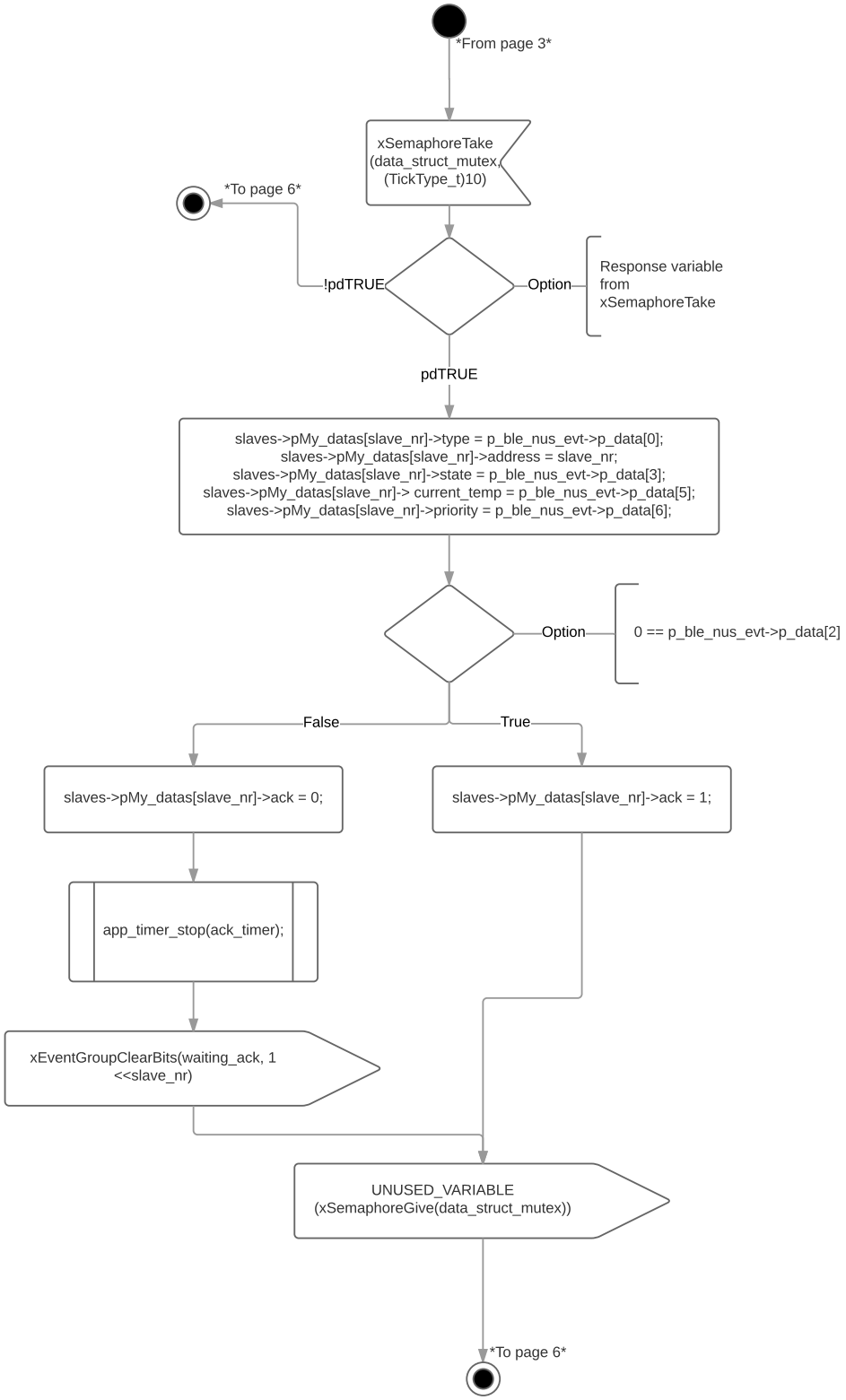
7.7.6 Function: ble_nus_c_evt_handler

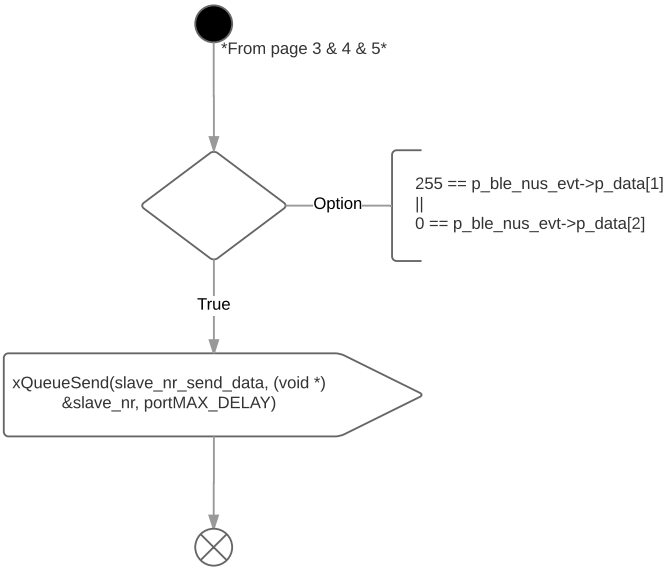



From page 1
Option
p_ble_nus_evt->evt_type
BLE_NUS_C_EVT_DISCOVERY_COMPLETE
BLE_NUS_C_EVT_NUS_RX_EVT
BLE_NUS_C_EVT_DISCONNECTED
err_code =
ble_nus_c_handlel_assign
(p_ble_nus_c,
p_ble_nus_evt->conn_handle,
&p_ble_nus_evt->handles);
adv_scan_start():
APP_ERROR_CHECK(err_code);
err_code =
ble_nus_c_rx_notif_enable
(p_ble_nus_c);
APP_ERROR_CHECK(err_code);
err_code = bsp_indication_set
(BSP_INDICATE_RCV_OK);
To page 3

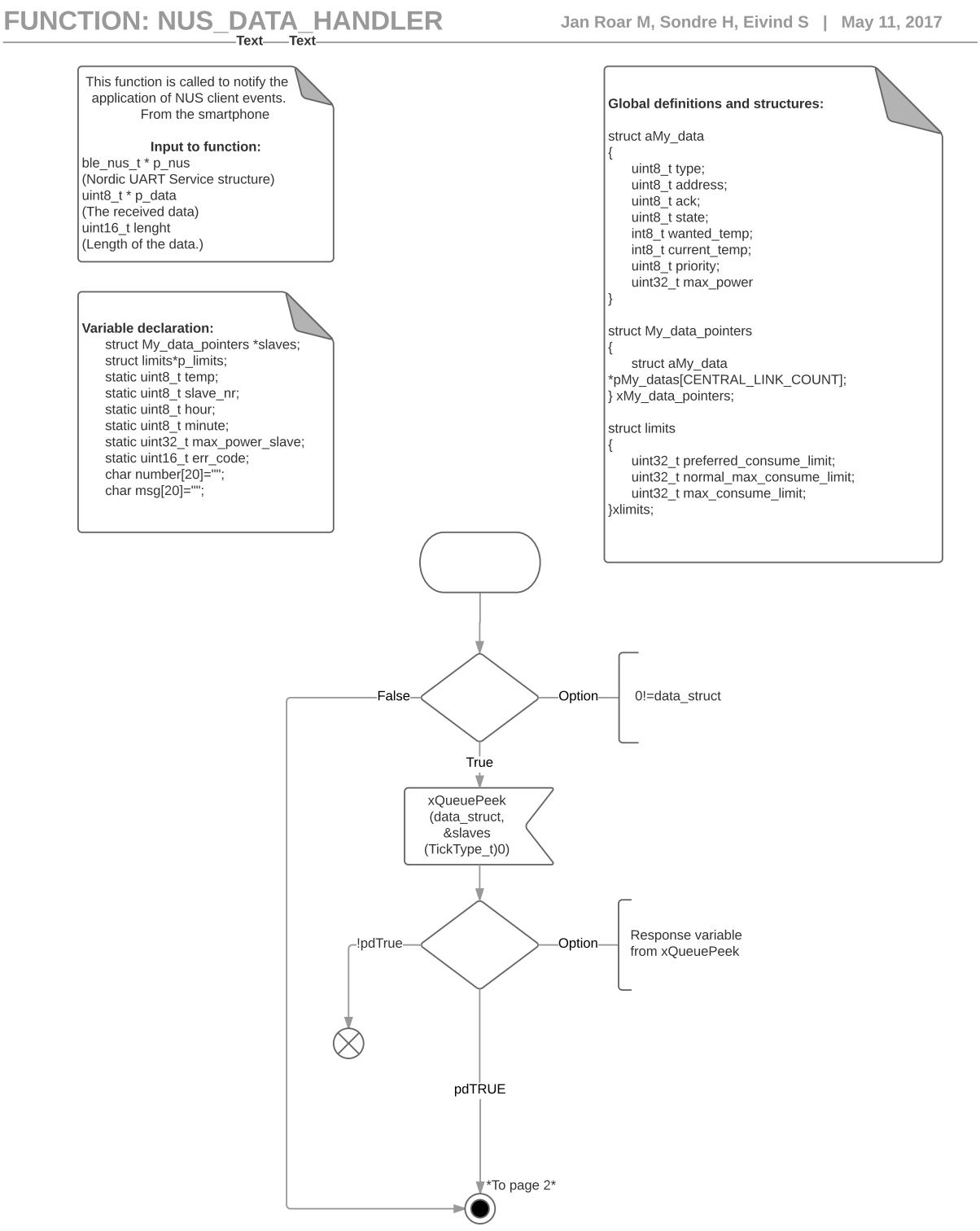

From page 2
Option
p_ble_nus_evt->p_data[0]
To page 6
'C'
To page 5
'b' || 'c' || 'D' || 'd' || default
'B'
xSemaphoreTake (data_struct_mutex, (TickType_t)10)
!pdTRUE
Option
Response variable from xSemaphoreTake
pdTRUE
False
Option
slaves->pMy_datas[slave_nr] ->priority != p_ble_nus_evt ->p_data[6]
&&
p_ble_nus_evt->p_data[6] <=19
True
slaves->pMy_datas[slave_nr]->priority = p_ble_nus_evt->p_data[6];
xSemaphoreGive(slave_on_bin_semaphore)
Option
Response variable from xSemaphoreGive
pdTRUE
To page 4

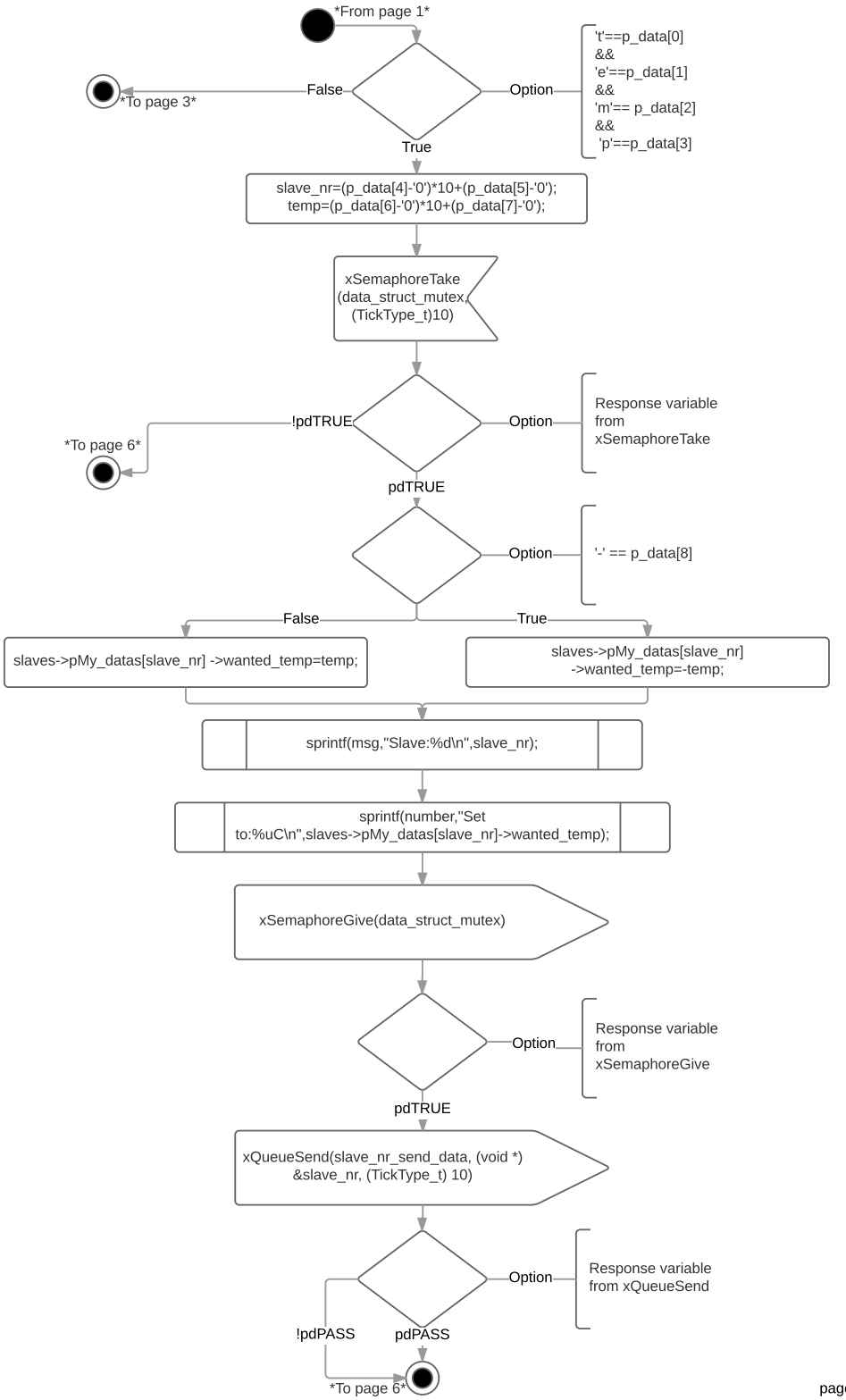

From page 3
slaves->pMy_datas[slave_nr]->type = p_ble_nus_evt->p_data[0];
slaves->pMy_datas[slave_nr]->address = slave_nr;
slaves->pMy_datas[slave_nr]->state = p_ble_nus_evt->p_data[3];
slaves->pMy_datas[slave_nr]-> current_temp = p_ble_nus_evt->p_data[5];
slaves->pMy_datas[slave_nr]->priority = p_ble_nus_evt->p_data[6];
Option
0 == p_ble_nus_evt->p_data[2]
False
True
slaves->pMy_datas[slave_nr]->ack = 0;
slaves->pMy_datas[slave_nr]->ack = 1;
app_timer_stop(ack_timer);
xEventGroupClearBits(waiting_ack, 1 <<slave_nr)
UNUSED_VARIABLE (xSemaphoreGive(data_struct_mutex))
To page 6

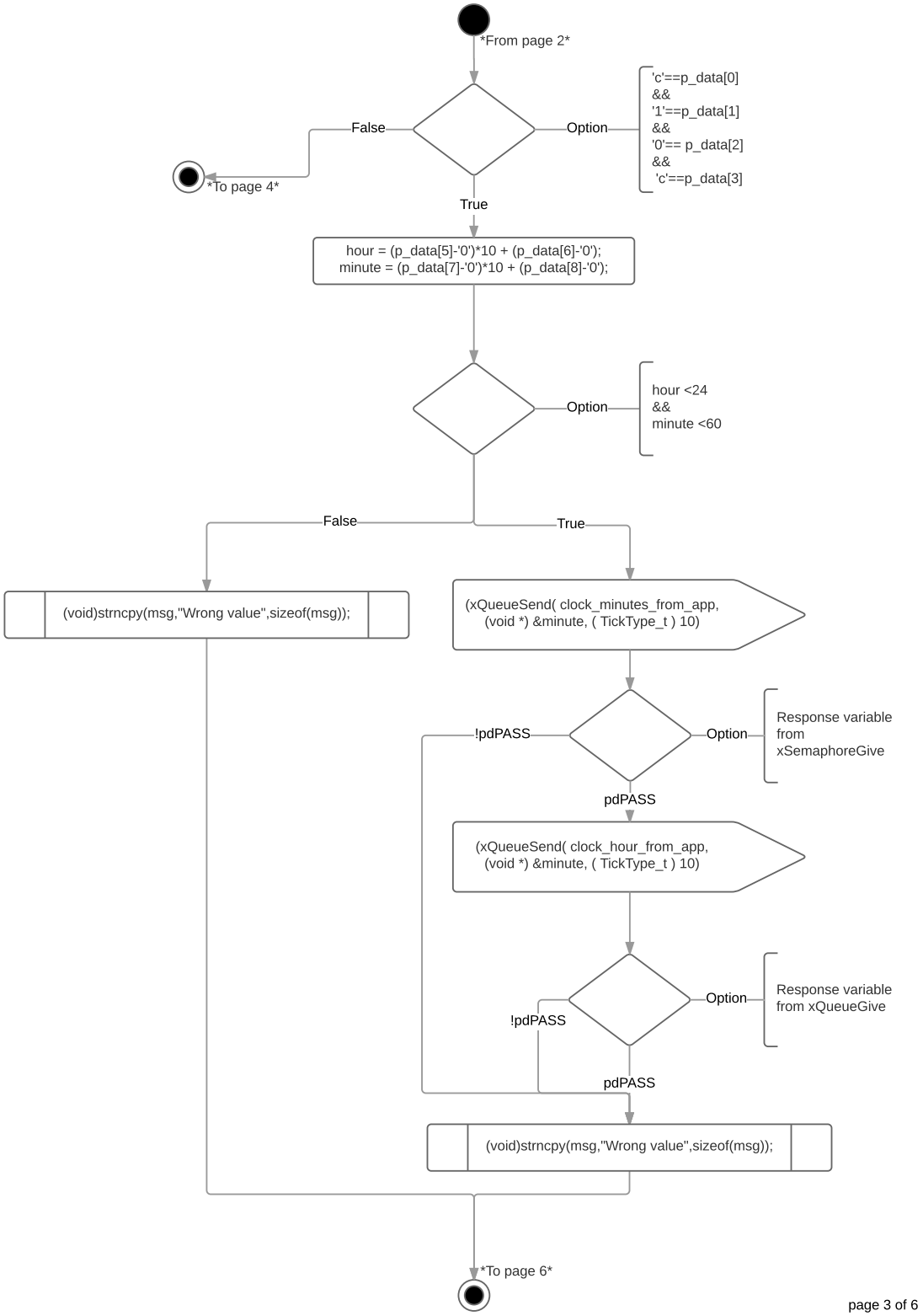

From page 3
xSemaphoreTake
(data_struct_mutex,
(TickType_t)10)
To page 6
!pdTRUE
Option
Response variable
from
xSemaphoreTake
pdTRUE
slaves->pMy_datas[slave_nr]->type = p_ble_nus_evt->p_data[0];
slaves->pMy_datas[slave_nr]->address = slave_nr;
slaves->pMy_datas[slave_nr]->state = p_ble_nus_evt->p_data[3];
slaves->pMy_datas[slave_nr]-> current_temp = p_ble_nus_evt->p_data[5];
slaves->pMy_datas[slave_nr]->priority = p_ble_nus_evt->p_data[6];
Option
0 == p_ble_nus_evt->p_data[2]
False
True
slaves->pMy_datas[slave_nr]->ack = 0;
slaves->pMy_datas[slave_nr]->ack = 1;
app_timer_stop(ack_timer);
xEventGroupClearBits(waiting_ack, 1
<<slave_nr)
UNUSED_VARIABLE
(xSemaphoreGive(data_struct_mutex))
To page 6

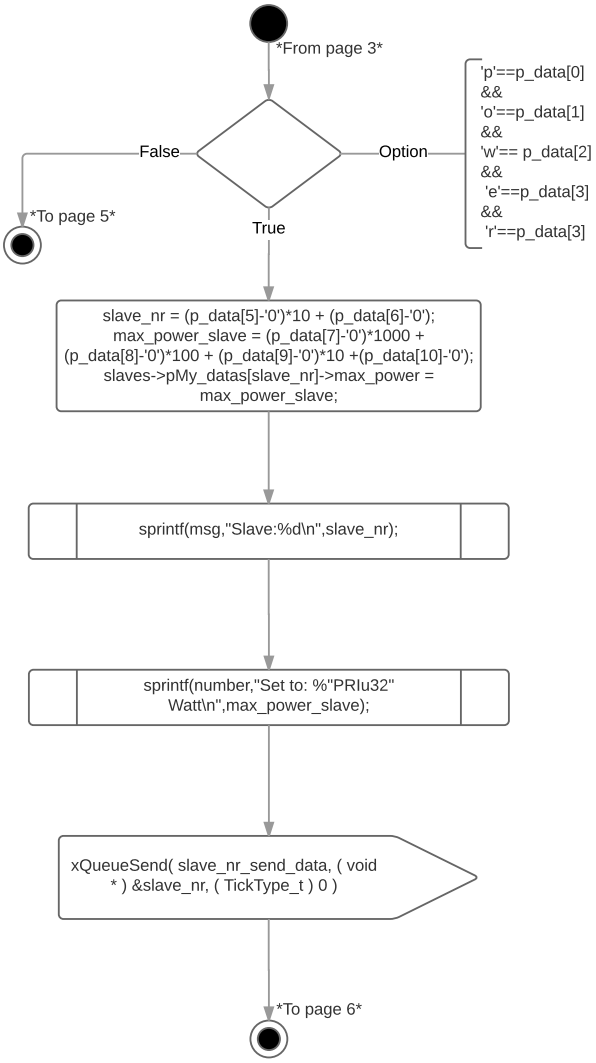

From page 3 & 4 & 5
Option
255 == p_ble_nus_evt->p_data[1]
||
0 == p_ble_nus_evt->p_data[2]
True
xQueueSend(slave_nr_send_data, (void *)
&slave_nr, portMAX_DELAY)

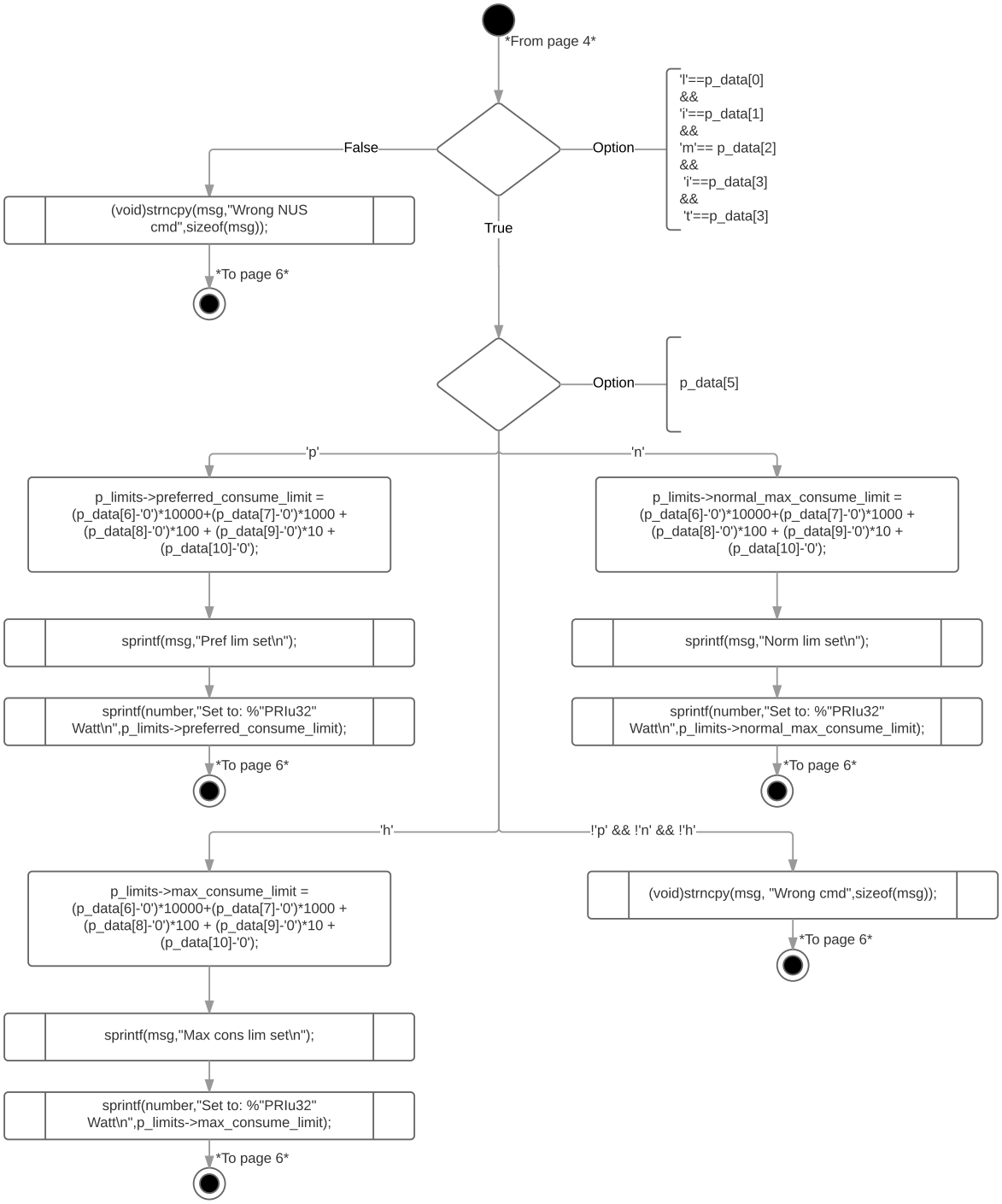
7.7.7 Function: nus_data_handler

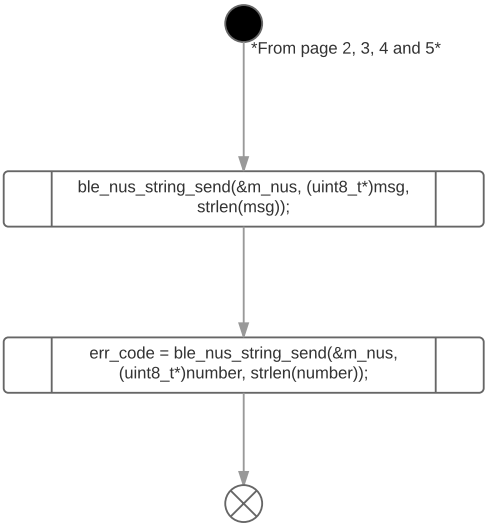



From page 1
False
Option
't'==p_data[0]
&&
'e'==p_data[1]
&&
'm'== p_data[2]
&&
'p'==p_data[3]
To page 3
True
slave_nr=(p_data[4]-'0')*10+(p_data[5]-'0');
temp=(p_data[6]-'0')*10+(p_data[7]-'0');
xSemaphoreTake
(data_struct_mutex,
(TickType_t)10)
!pdTRUE
Option
Response variable
from
xSemaphoreTake
To page 6
pdTRUE
Option
'-' == p_data[8]
False
True
slaves->pMy_datas[slave_nr] ->wanted_temp=temp;
slaves->pMy_datas[slave_nr]
->wanted_temp=-temp;
sprintf(msg,"Slave:%d\n",slave_nr);
sprintf(number,"Set
to:%uC\n",slaves->pMy_datas[slave_nr]->wanted_temp);
xSemaphoreGive(data_struct_mutex)
Option
Response variable
from
xSemaphoreGive
pdTRUE
xQueueSend(slave_nr_send_data, (void *)
&slave_nr, (TickType_t) 10)
Option
Response variable
from xQueueSend
!pdPASS
pdPASS
To page 6


From page 2
'c'==p_data[0]
&&
'1'==p_data[1]
&&
'0'== p_data[2]
&&
'c'==p_data[3]
Option
False
To page 4
True
hour = (p_data[5]-'0')*10 + (p_data[6]-'0');
minute = (p_data[7]-'0')*10 + (p_data[8]-'0');
hour <24
&&
minute <60
Option
False
True
(void)strncpy(msg,"Wrong value",sizeof(msg));
(xQueueSend(clock_minutes_from_app,
(void *) &minute, (TickType_t) 10)
!pdPASS
Option
Response variable
from
xSemaphoreGive
pdPASS
(xQueueSend(clock_hour_from_app,
(void *) &minute, (TickType_t) 10)
Option
Response variable
from xQueueGive
!pdPASS
pdPASS
(void)strncpy(msg,"Wrong value",sizeof(msg));
To page 6


From page 3
'p'==p_data[0]
&&
'o'==p_data[1]
&&
'w'== p_data[2]
&&
'e'==p_data[3]
&&
'r'==p_data[3]
Option
False
True
To page 5
slave_nr = (p_data[5]-'0')*10 + (p_data[6]-'0');
max_power_slave = (p_data[7]-'0')*1000 +
(p_data[8]-'0')*100 + (p_data[9]-'0')*10 +(p_data[10]-'0');
slaves->pMy_datas[slave_nr]->max_power =
max_power_slave;
sprintf(msg,"Slave:%d\n",slave_nr);
sprintf(number,"Set to: %"PRIu32"
Watt\n",max_power_slave);
xQueueSend(slave_nr_send_data, (void
*) &slave_nr, (TickType_t) 0)
To page 6


From page 4
'l'==p_data[0]
&&
'i'==p_data[1]
&&
'm'== p_data[2]
&&
'i'==p_data[3]
&&
't'==p_data[3]
Option
False
True
(void)strncpy(msg,"Wrong NUS cmd",sizeof(msg));
To page 6
Option
p_data[5]
'p'
'n'
p_limits->preferred_consume_limit = (p_data[6]-'0')*10000+(p_data[7]-'0')*1000 + (p_data[8]-'0')*100 + (p_data[9]-'0')*10 + (p_data[10]-'0');
p_limits->normal_max_consume_limit = (p_data[6]-'0')*10000+(p_data[7]-'0')*1000 + (p_data[8]-'0')*100 + (p_data[9]-'0')*10 + (p_data[10]-'0');
sprintf(msg,"Pref lim set\n");
sprintf(msg,"Norm lim set\n");
sprintf(number,"Set to: %"PRIu32" Watt\n",p_limits->preferred_consume_limit);
sprintf(number,"Set to: %"PRIu32" Watt\n",p_limits->normal_max_consume_limit);
To page 6
To page 6
'h'
!'p' && !'n' && !'h'
p_limits->max_consume_limit = (p_data[6]-'0')*10000+(p_data[7]-'0')*1000 + (p_data[8]-'0')*100 + (p_data[9]-'0')*10 + (p_data[10]-'0');
(void)strncpy(msg, "Wrong cmd",sizeof(msg));
To page 6
sprintf(msg,"Max cons lim set\n");
sprintf(number,"Set to: %"PRIu32" Watt\n",p_limits->max_consume_limit);
To page 6


From page 2, 3, 4 and 5
ble_nus_string_send(&m_nus, (uint8_t*)msg, strlen(msg));
err_code = ble_nus_string_send(&m_nus, (uint8_t*)number, strlen(number));

7.7.8 Header file: m_bus_receiver.h

D:\Programming\nRF5_SDK_12.2.0_f012efa\examples\ble_peripheral\ble_freertos_bachelor_work_on_this_one\m_bus_receiver.h

```
/** @file
 *
 * @defgroup m_bus_receiver
 *
 * @brief   Used in connection with m_bus power meter
 *
 * @details This header file contains different functions regarding the connection between a micro-controller and
 *          the m_bus power meter. The different connections is initializing,  changing the m-bus receivers primary address,
 *          and also resetting the partial power of the m_bus
 *          This also features different checks.
 *
 * @note   Some of the function should be protected by a mutex when used in a application with a RTOS type
 *
 */
#ifndef M_BUS_RECEIVER
#define M_BUS_RECEIVER

#include <stdint.h>
#include <stdbool.h>
#include <string.h>

#define MAXIMUM_ADDRS 250

/*
Initialisation definitions
*/
#define START_INIT 0x10                     //Start field value
#define SEND_OR_REPLY_INIT 0x40             //Send or reply, reset- field value
#define STOP_INIT 0x16                      //Stop field value.

#define TOTAL_LENGTH_INIT_FRAME 0x05        //Total length of the init telegram
#define RESPONSE_INIT 0XE5                  //Response from m_bus after initialisation

/*
Changing primary address definitions
*/
#define START_CHANGING_ADDR 0x68            //Start field value
#define FIELD_LENGHT_CHANGING_ADDR 0x06     //Field Length value
#define C_FIELD_CHANGING_ADDR 0x68          //C field value
#define CI_FIELD_CHANGING_ADDR 0x51         //CI field value
#define DIF_FIELD_CHANGING_ADDR 0x01        //DIF field value
#define VIF_FIELD_CHANGING_ADDR 0x7A        //VIF field value
#define STOP_FIELD_CHANGING_ADDR 0x16       //Stop field value

#define TOTAL_LENGTH_CHANGING_ADDR 0x0C     //Total length of the changing primary address telegram
#define RESPONSE_CHANGING_ADDR 0xE5         //Response from m_bus after changing the primary address.

/*
Reset ACC(application reset)
*/
#define START_RESET_ACC 0x68                //Start field value
#define FIELD_LENGTH_RESET_ACC 0x03         //Field length value
#define C_FIELD_RESET_ACC 0x53              //C field value
#define CI_FIELD_RESET_ACC 0x50             //CI field value
#define STOP_FIELD_RESET_ACC 0x16           //Stop field value

#define TOTAL_LENGTH_RESET_ACC 0x09         //Total length of the application reset telegrame
#define RESPONSE_RESET_ACC 0xE5             //Response from m_bus after application reset

/*
Reset total partial power
*/
#define START_RESET_PARTIAL 0x68            //Start field value
#define FIELD_LENGTH_RESET_PARTIAL 0x04     //Field length value
#define C_FIELD_RESET_PARTIAL 0x53          //C field value
#define CI_FIELD_RESET_PARTIAL 0x50         //CI field value
#define RESET_COUNTER_PARTIAL 0x01          //Reset counter field value
#define STOP_FIELD_RESET_PARTIAL 0x16       //Stop field value

```

Page 1

D:\Programming\nRF5_SDK_12.2.0_f012efa\examples\ble_peripheral\ble_freertos_bachelor_work_on_this_one\m_bus_receiver.h

```
70  #define TOTAL_LENGTH_RESET_PARTIAL 0x0A     //Total length of the partial power reset telegram
71  #define RESPONSE_RESET_PARTIAL 0xE5         //Response from m_bus after partial power reset.
72
73  /*
74  REQ_UD2: Definitions for requesting data from m_bus receiver
75  */
76  #define START_REQUEST 0x10                  //Start field value
77  #define C_FIELD_FCB_NOT_SET_REQUEST 0x5B    //C field value with FCB bit set to 0
78  #define C_FIELD_FCB_SET_REQUEST 0x7B        //C field value with FCB bit set to 1
79  #define STOP_REQUESET 0x16                  //Stop field value
80
81  #define TOTAL_LENGHT_REQUEST 0x05           //Totalt length of the REQ_UD2 telegram.
82  //Response from REQ_UD2 is a 62 bytes telegram from the m_bus receiver.
83
84  /*
85  Definition used for response telegram from m_bus after REQ_UD2 reguest.
86  */
87  #define RESPONSE_START_FIELD 0x68           //1 and 4 byte in the telegram
88  #define RESPONSE_L_READ 0x38                //2 and 3 byte in the telegram
89  #define RESPONSE_STOP_FIELD 0x16            //62 byte in the telegram
90
91
92  /*
93  States used in the uart_search_thread
94  */
95  typedef enum
96  {
97    SEARCING_NEW_ADR,
98    WAITING_RESPONSE_STATE,
99    CREATE_SEND_REQ_TASK,
100   WAITING_STATE
101 } uart_event_states;
102
103 /*
104 States used in the uart_thread
105 */
106 typedef enum
107 {
108   TIMER_ACTIVE_OR_NOT,
109   SENDING_REQUD2,
110   READING_RESPONSE
111 } uart_reading_states;
112
113
114 /*
115 Structure to save the values from the m_bus receiver. (currently not in use)
116 */
117 struct aMessage
118 {
119   uint32_t Message_number;
120   uint8_t adr;
121   uint8_t STAT;
122   uint32_t Total_power;
123   uint32_t Partial_power;
124   uint16_t Voltage;
125   uint16_t Current;
126   uint16_t Power;
127   uint16_t Reactive_power;
128 } xMessage;
129
130 /*
131 Structure to hold address of m_bus and number of adrs.
132 */
133 typedef struct adr_of_m_bus_struct
134 {
135   uint8_t adr_array[10];
136   uint8_t number_of_adrs;
137 } adr_struct;
138
139 /**@brief Function for initializing the m-bus receiver
140  *
141  * @details This should be protected by a mutex when using a RTOS.
```

D:\Programming\nRF5_SDK_12.2.0_f012efa\examples\ble_peripheral\ble_freertos_bachelor_work_on_this_one\m_bus_receiver.h

```
142     *
143     *
144     *
145     * @param[in]   adr_off_m_bus      The adresse to the m_bus receiver
146     *
147     */
148    void m_bus_receiver_init(uint8_t adr_off_m_bus);
149
150
151    /**@brief Function for changing the m-bus receivers primary address
152     *
153     * @details This should be protected by a mutex when using a RTOS.
154     *
155     *
156     *
157     * @param[in]   primary_adr_off_m_bus      The primary adress off the m-bus receiver
158     * @param[in]   new_address_off_m_bus      The new adress that we want to set the m-bus receiver to
159     *
160     */
161    void m_bus_receiver_changing_primary_address(uint8_t primary_adr_off_m_bus, uint8_t
       new_address_off_m_bus);
162
163
164    /**@brief Function for resetting the m-bus receiver
165     *
166     * @details This should be protected by a mutex when using a RTOS.
167     *
168     *
169     *
170     * @param[in]   primary_adr_off_m_bus      The primary adress off the m-bus receiver
171     *
172     */
173    void m_bus_receiver_reset_application(uint8_t primary_adr_off_m_bus);
174
175
176    /**@brief Function for resetting the m-bus receiver partial power
177     *
178     * @details This should be protected by a mutex when using a RTOS.
179     *
180     *
181     *
182     * @param[in]   primary_adr_off_m_bus      The primary adress off the m-bus receiver
183     *
184     */
185    void m_bus_receiver_reset_partial_power(uint8_t primary_adr_off_m_bus);
186
187
188    /**@brief Function for requesting telegram from the uart module (REQ_UD2)
189     *
190     * @details M_bus_receiver requires a REQ_UD2 query in order to send response. The response is a RSP_UD
       telegram 62 bytes.
191     *          This should be protected by a mutex when using a RTOS.
192     *
193     *
194     * @param[in] adr_off_m_bus    The adress of the m_bus_receiver that we want response from.
195     * @param[in] c_field          The c-field, with or without fcb set.
196     *
197     */
198     void m_bus_send_request(uint8_t primary_adr_off_m_bus, uint8_t c_field);
199
200
201     /**@brief Function for checking response from the m_bus_receiver
202     *
203     * @details Response from the m_bus_receiver should be 0xE5 after when sending everything else then
       REQ_UD2
204     *
205     *
206     *
207     * @param[in] exp_response    The expected response from the m_bus receiver
208     *
209     * @retval True if the response from m_bresponse_from_m_busus_receiver is 0xE5 (#define RESPONSE 0xE5)
210     */
```

D:\Programming\nRF5_SDK_12.2.0_f012efa\examples\ble_peripheral\ble_freertos_bachelor_work_on_this_one\m_bus_receiver.h

```
211  bool response_from_m_bus (uint8_t exp_response);
212
213   /**@brief Function for checking that the start of telegram is correct
214    *
215    * @details Checking that the first 4 bytes of a RSP_UP is correct.
216    *          byte0 == RESPONSE_START_FIELD
217    *          byte1 == RESPONSE_L_READ
218    *          byte2 == RESPONSE_L_READ
219    *          byte3 == RESPONSE_START_FIELD
220    *
221    * @param[in] byte0   First byte of the RSP_UD
222    * @param[in] byte1   Second byte of the RSP_UD
223    * @param[in] byte2   Third byte of the RSP_UD
224    * @param[in] byte3   Fourth byte of the RSP_UD
225    *
226    * @retval True is the 4 first byte is correct
227    */
228  bool telegram_structure_check (uint8_t byte0, uint8_t byte1, uint8_t byte2, uint8_t byte3);
229
230
231   /**@brief Function for decoding bcd.
232    *
233    * @details Takes in a bcd decoded byte and turn it in to a integer
234    *
235    *
236    *
237    * @param[in] The value that is coded in bcd
238    *
239    * @retval    The value in a uint type
240    */
241  uint8_t bcdtobyte(uint8_t bcd);
242
243
244  /**@brief Function to check for correct checksum
245   *
246   * @details  Calculates the checksum by adding c_field, adr and ci_field and compares this with the
     expected checksum
247   *
248   * @param[in] c_field       The c-field, with or without fcb set
249   * @param[in] adr           Address of the m_bus meter
250   * @param[in] ci_field      CI field
251   * @param[in] checksum      Expected checksum
252   *
253   * @retval    True if the expected cheksum is the same as the calculated
254   */
255  bool correct_checksum(uint8_t c_field, uint8_t adr, uint8_t ci_field, uint8_t checksum);
256
257
258  #endif //M_BUS_RECEIVER
259
```