AC Sensor Project

Placement Report

24 April 2017

Executive Summary

With the increased demand for energy across the rapidly globalising world draws a need for ways to improve energy monitoring systems. For grid utilities, and similarly for off-grid energy providers, all these energy providers require a low-cost and accurate method of metering the energy provided to their users for billing purposes. Furthermore, these is also a need to improve the energy monitoring of these systems to utilise data analytics and improve the efficiency of the systems with appropriate load management.

With the advent of cloud monitoring systems and internet of things devices comes the opportunity to lower the costs of metering devices. With BBOXX having the technology for GSM connectivity to a cloud-based server, the remaining task would be to design an AC meter. This report aims to produce a smart meter that does all of the above: to design a low-cost and accurate Smart AC Meter with the means of transmitting data to a cloud-based system for further data analysis.

The scope for existing AC Meters is varied, with very few providing GSM connectivity for rural areas, while most are still running electromechanical meters, which are bulky and requires manual readings. The solution proposed in this report provides a fully integrated, electronic meter with an easy to use interface for readings. Furthermore, the overall cost of the AC Meter proposed is lower than that of market options.

The design for the AC Meter consists of several layers: The conceptual stage, the hardware design stage, the prototyping stage, the firmware stage and lastly, the manufacture stage. The conceptual stage consists of a look at current market options, and deciding the best possible method with regards to costs and accuracy for the sensing method. The hardware design stage, for which the initial specifications are determined, and the key components are tested and chosen. The prototyping stage, for which the hardware design is manufactured and finalised for the sensors, microcontroller and external communications network. The firmware stage, for which the controls for the sensor is done and the data is processed before being sent. Lastly, the manufacture stage, which deals with the large scale manufacturing of the device.

Smart AC Metering is an emerging market for which the demand is expected to grow, with Smart Meters in the UK to be the standard In the UK by 2020[1]. The overall cost of the device, with the functionalities provided, makes it a useful option for Smart Meters in the future, and provides an advantageous position for BBOXX to enter the Smart Metering market.

[1] https://www.gov.uk/guidance/smart-meters-how-they-work

Table of Contents

1. Introduction

This report provides an insight into the technical and personal experiences reaped during the 6-month industrial placement at BBOXX Ltd, and details the design processes from the conceptual phase to the manufacturing phases. The report is structured as such: A comprehensive view of the BBOXX business and product history would be given as a background into the current projects undertaken, and this will be elaborated on in **Section 2. Company Overview**. The aims and scopes of the different projects would be stated in **Section 3**. **Project Aims, Plans and Approaches**, with the deliverables stated and strategy of approach shown via means of a breakdown of tasks and a timeline for the entire 6-months displayed in a Gantt chart. An in-depth account would then be specified for the work done during these 6 months in **Section 4.Details of Work**, and finally a reflection of the experiences garnered during the time here will be given in **Section 5. Reflections**.

1.1 BBOXX Overview

BBOXX is a venture-backed company developing solutions for providing affordable, clean energy to off-grid communities in the developing world. Their customers tend to consist of farmers or fishermen living in rural areas in Kenya or Rwanda who make around $100-$200 a month.

Previously, people in these areas used to spend around $8 - $12 a month on kerosene or dry batteries to charge their phones. BBOXX provides a solution by offering solar power, and is novel in its approach by providing financing for the customers. This financing approach allows for a larger customer base, and will be further covered in **Section 2**.

1.2 Placement Role

My role through this placement was as a hardware and embedded systems engineer in London, where I was responsible for the development of an AC Meter. In this position I was exposed to power engineering concepts, digital electronics, communications networks and protocols, real-time digital signal processing and embedded systems, while also gaining experience in programming for C and Python.

1.3 Placement Projects

The main project I was involved in was the AC Meter project during my 6 months placement. This project occupied most of my time during this placement, as I was responsible for the initial research, design and planning for the entire project. In order to determine the project requirements, it was essential for me to understand the BBOXX business processes, from the design phases in London, to the manufacturing phases in China, and to the distribution phases in Africa. This led to the initial plans I drafted for the project, which was supervised by my fellow colleagues.

2 side projects I was given was to create a 50A current sensor for larger currents, and to also create a documentation system to ensure that the projects I was working on could be easily handed over to new employees.

1.4 Personal Experience

Creating a product from scratch provided an immense satisfaction for me. The process was definitely not easy, as it required an understanding into many different fields all at once. I had to learn new software tools, trudge through google to look for different open-source options, draw out and test different schematics, figure out a new way to send a communications protocol, and lastly to utilise systems which I was not used to(Linux and Command Line Interfaces.

However, given the responsibility of a project and having had the chance to create something new gave me the drive to keep learning, with the ultimate goal being a device that had my unique style and footprint. The process of designing this product

allowed me to garner priceless experience for embedded systems engineering, and also gave me a new insight into hardware engineering.

2. Company Overview

2.1 BBOXX LTD

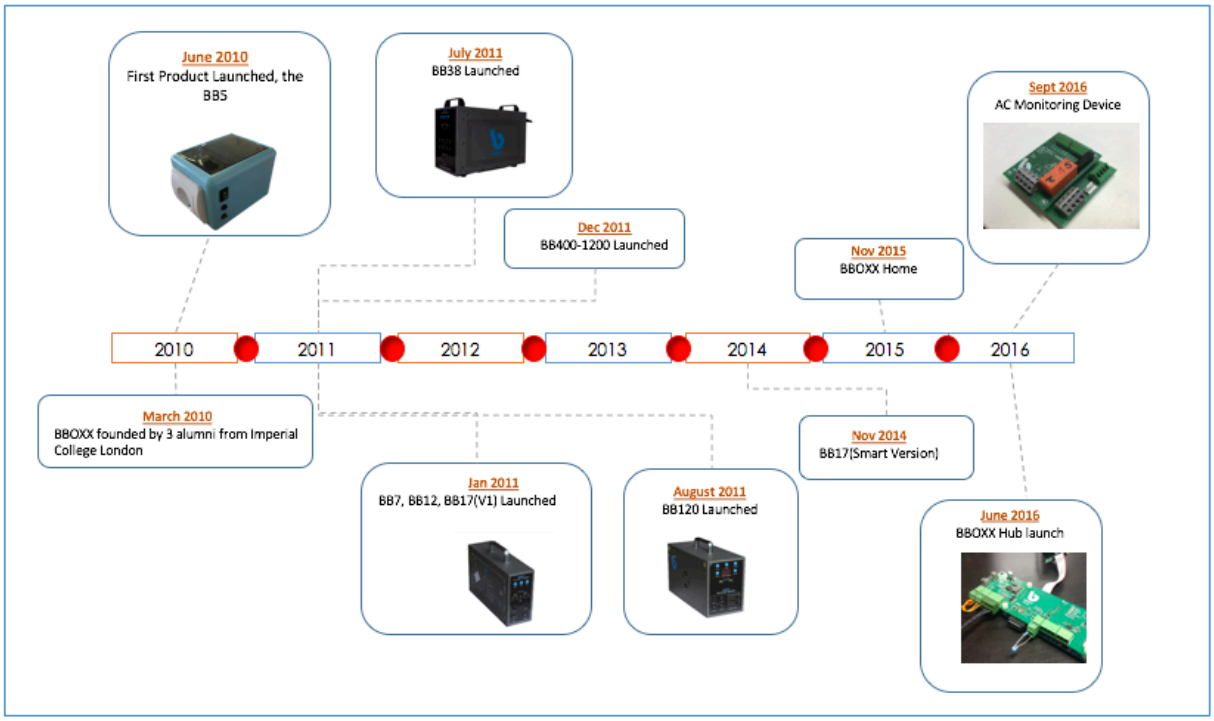

Figure 2.1: Timeline of BBOXX Products from 2010 - 2016

2.1.1 Company Overview

BBOXX was founded in March 2010 by three alumni from the EEE Department in Imperial College London. The aim of the company is to provide 20 million people with electricity by 2020, and are currently distributing in Rwanda, Kenya and Pakistan.

The company operates in 3 different locations. Firstly, the engineering team, which are mainly based in London and are in charge of the BBOXX servers and hardware design. Secondly, the manufacturing team based in China which deals with manufacturing and testing of products. Lastly, the operations team, which is based in Kigali and deals with the distribution and sales of the products in the region. **Figure 2.1** displays the timeline of BBOXX Products from 2010 to 2016, and this will be important in explaining the current projects being done.

2.1.2 Product Evolution

The initial BBOXX products in 2010 and 2011(**BB5** to **BB1200**) began as a plug-and-play system. Customers bought a system consisting of a solar panel and a battery which included DC outputs for lighting and charging. BBOXX would supply these products to local distributors, which managed the distribution of these standalone products.

However, this product cost around $400 dollars, which put it out of the range of many of the customers in Rwanda and Kenya. In order to facilitate scaling, BBOXX required a financing structure for customers to pay in instalments. This allowed customers to pay via $20 instalments every month, which was affordable for more customers, thus increasing the user-base.

A by-product of this financing structure was to build up a method of monitoring these products and be able to switch them off remotely in the case where customers stopped paying their instalments, or when their batteries were failing and required replacing. This resulted in the development of the **SMART Solar**, a server-based system that allowed BBOXX to remotely monitor the products. This was developed for 2 years, which led to the development of the **BB17-Smart** and eventually involved into their flagship product, the **BBOXX Home.**

As shown in **Figure 2.2**, the **SMART Solar Dashboard** allows for remote monitoring and battery management of the systems. The usage of each device is stored and sent via GSM to a BBOXX cloud-based server, where data analysis is done in the backend to extend the battery life based on real-time data. This system also provides the location of each BBOXX device (with accuracy up to 1km) for cases where appliances have to be repossessed due to customers not following up on payments.

This **SMART Solar Dashboard** system was essential in providing a competitive edge to BBOXX that many other companies did not have. Furthermore, the financing structure provided an entire new customer base to BBOXX.

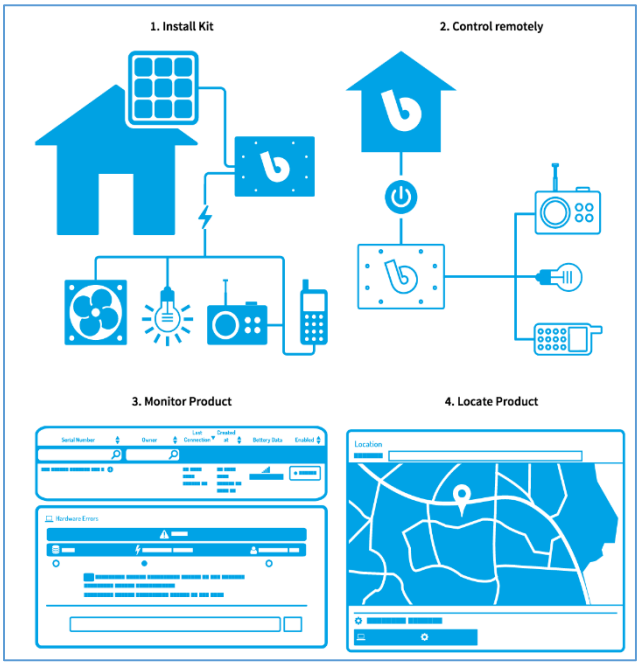

Figure 2.2: Smart Solar Dashboard

2.1.3 Current Products and Role

With the success of the **Smart Solar Dashboard**, and having have multiple units (30,000 as of August 2016) functioning well in the field, the next step for the company was to utilize the demand for remote monitoring to other systems. Off-grid energy providers required a solution for managing their grid systems, and needed a tested and reliable method to monitor the energy being metered to their customers. This led to interest for BBOXX to develop an AC metering device that could be remotely monitored as well.

The latest BBOXX product, the **BBOXX Hub**, represents a new era for the BBOXX business. With the demand for Internet of Things devices, where devices are now embedded with network connectivity with them also being able to interconnect and exchange data, the **BBOXX Hub** represented a viable product with a sufficient customer base. The **BBOXX Hub** consists of the GSM hardware capable of interfacing with the SMART Solar API instructions, and also communications management systems for interfacing with the different protocols for data transmission such as RS485, Inverter Control, and DC/AC Power Lines. **Annex A** shows the different configurations available.

This led to the requirement for a dedicated AC Meter, which worked in tandem with the **BBOXX Hub**. As mentioned earlier, my role involves developing the AC Meter for BBOXX, and to present ideas and solutions faced for the development of this product.

Figure 2.3: BBOXX Hub

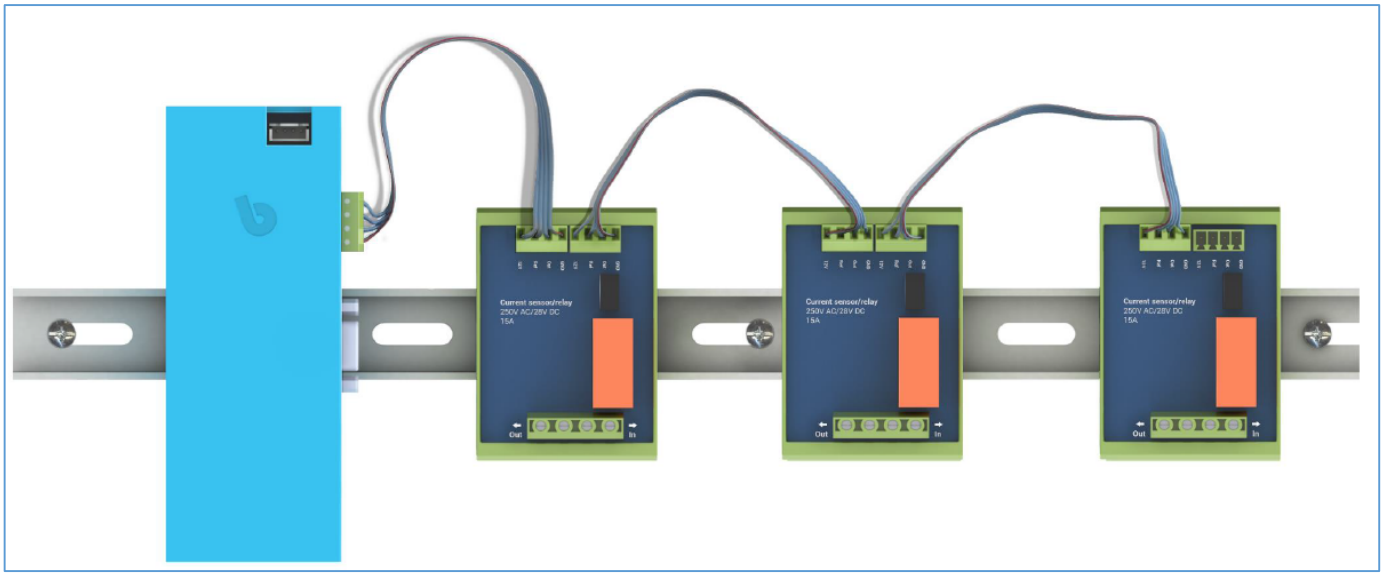

Figure 2.4: Vision of multiple AC Sensors communicating with the BBOXX Hub

Figure 2.4 displays the vision held for the AC Sensors and the BBOXX Hub. This would be my role for this placement – To design an implement the AC Sensor for communications with the BBOXX Hub, and to make this vision a working reality.

3. Project Aims, Plans and Approaches

3.1 AC Sensor Project

3.1.1 Aims

The aim of this project **is to implement an AC Meter for use with the BBOXX Hub**. To implement the meter, a multi-stage approach was favoured in order to break it down into manageable phases. The plan I decided upon consisted of determining the desired deliverables for each stage of the process, before planning the timeline for all these deliverables to be achieved.

Deliverables

Due to my relative inexperience as a hardware engineer, I wanted to plan for a sufficient timeframe for each stage to ensure that the difficulty was progressive. For example, the manufacturing stage was something which required a while to accomplish, as I needed to have a clear idea of BBOXX processes if I wanted to plan for that stage. Hence, I shifted it to the last stage as I felt I needed the time to have a better understanding.

Conceptual Stage

The conceptual stage involved market research and a thorough investigation into different types of technologies available for use. The deliverables for this stage are as follows:

1. Determine if the AC Meter market is worth entering.
2. Determine the initial specifications required for the AC Meter, and decide on the best possible method for sensing the readings.

Hardware Design Stage

The Hardware design stage consists of tests on the theoretical circuit with simulations and also by choosing individual components for testing. The deliverables for this stage are as follows:

1. Determine the schematic required for the AC Meter circuit.
2. Test and determine the key components required for the AC Meter circuit.

Prototyping Stage

The prototyping stage aims to finalise the design for the AC Meter, and takes into account the design considerations chosen in the previous stage to produce the prototypes. This stage would require more testing in order to ensure that the final product is robust and error-free.

1. Design the Schematic required for AC Meter.
2. Design the PCB required for the AC Meter.
3. Test, Review and Finalise design

Firmware Design Stage

The firmware design stage plans for the control flow of the program commands in the embedded systems, and also determines the communication protocols required for both internal and external communications.

1. Firmware Overview Plan
2. Coding individual modules before Integration.

Manufacturing Stage

The manufacturing stage takes into account that the design has been finalized, and the aim would be to design a test bench capable of programming and testing the AC Meter efficiently during the manufacturing stage.

1. Manufacturing Planning
2. Design of Components Required

3.2 50A Current Sensor

Aims

The aim of this project comprised of **designing a current sensor capable of monitoring DC current from 0-50A**. This was done during the main project and drew upon the designs made during the main project.

1. Design a 50A Current Sensor
2. Test and Produce Samples

3.3 Hardware Documentation

Aims

The aim of this project was to product a hardware documentation template for the ease of handover for future employees.This was an ongoing project that arose from the need to document hardware design choices made during the project. This was done in order to assist with the handover of projects in the future, and to also standardize documentation across the company.

1. Documentation Template

3.4 Project Deliverables with Approaches

The table below describes the different projects, with the subtasks required for each deliverable. This is done in order to separate the projects into distinct and manageable phases and proper timeline planning, which will be shown in **Section 3.4.**

AC Meter Project

Phase	Deliverables	Approaches Taken
Conceptual	Determine if the AC Meter market is worth entering.	(1) Market Research on current metering solutions
	Determine the initial specifications required for the AC Meter, and decide on the best possible method for sensing the readings.	(1) Build up initial specifications of AC Meter for design (2) Choose between different sensoring methods for AC voltage and current
Design	Determine the schematic required for the AC Meter circuit.	(1) Simulate circuits on LTSpice (2) Safety Measures (3) Detailed Specifications
	Test and determine the components required for the AC Meter circuit	(1) Obtain Evaluation boards and test separate components
Prototyping	Design the Schematic required for AC Meter.	(1) Gain proficiency in Altium Designer (2) Split schematic into different modules for ease of design
	Design the PCB required for the AC Meter.	(1) Understand how PCB layouts are done (2) Finalise PCB layout
	Test, Review and Finalise Design	(1) Rigorous Testing on PCB done (2) Redesign PCB for improved revisions
Firmware Design	Firmware Overview Plan	(1) Determine the Software Required (2) Plan for how the modules in the system communicate
	Coding individual Modules	(1) Determine the functional Requirements (2) Begin working on individual Modules (3) Put the code together (4) External Interrupts built in
Manufacturing	Manufacturing Planning	(1) Understand the requirements for programming the flash

		(2) Create a PCB for programming the senso
	Design the required components	(1) Create a PCB for programming the sensor

50A Current Sensor

Deliverables	Approaches Taken
Design a 50A Current Sensor	(1) Look for components and test them
Test and Produce Samples	(2) Produce the samples and ensure they undergo proper tests

Hardware Documentation

Deliverables	Approaches Taken
Document Hardware Template	(1) Continue logging everything done in a single day, and start writing what has been done in a word document before the end of each week.

3.5 Project Timeline

The forecasted project schedule from April to June 2016 is shown in **Figure 3.1,** while the project schedule for July to September 2016 is shown in **Figure 3.2**. The column on the right represents the deliverables to be met during the placement period, with them numbered accordingly. The status column displays whether a deliverable had been done, and whether a stage had been complete. A green flag indicates a completed stage, while a green tick indicates a completed deliverable. On the other hand, a red flag indicates an incomplete stage, while a red cross indicates an incomplete deliverable.

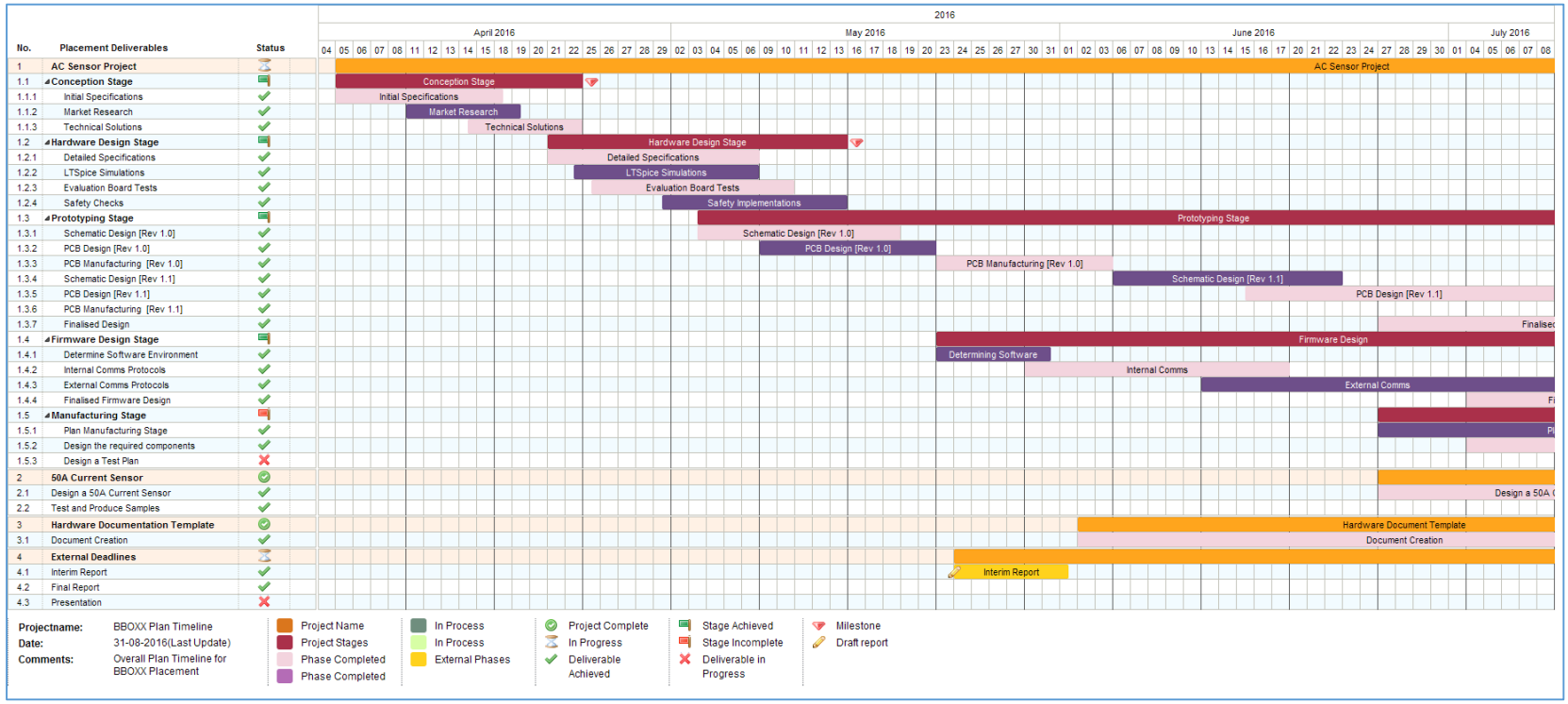

Figure 3.1: Project Schedule from April to June 2016

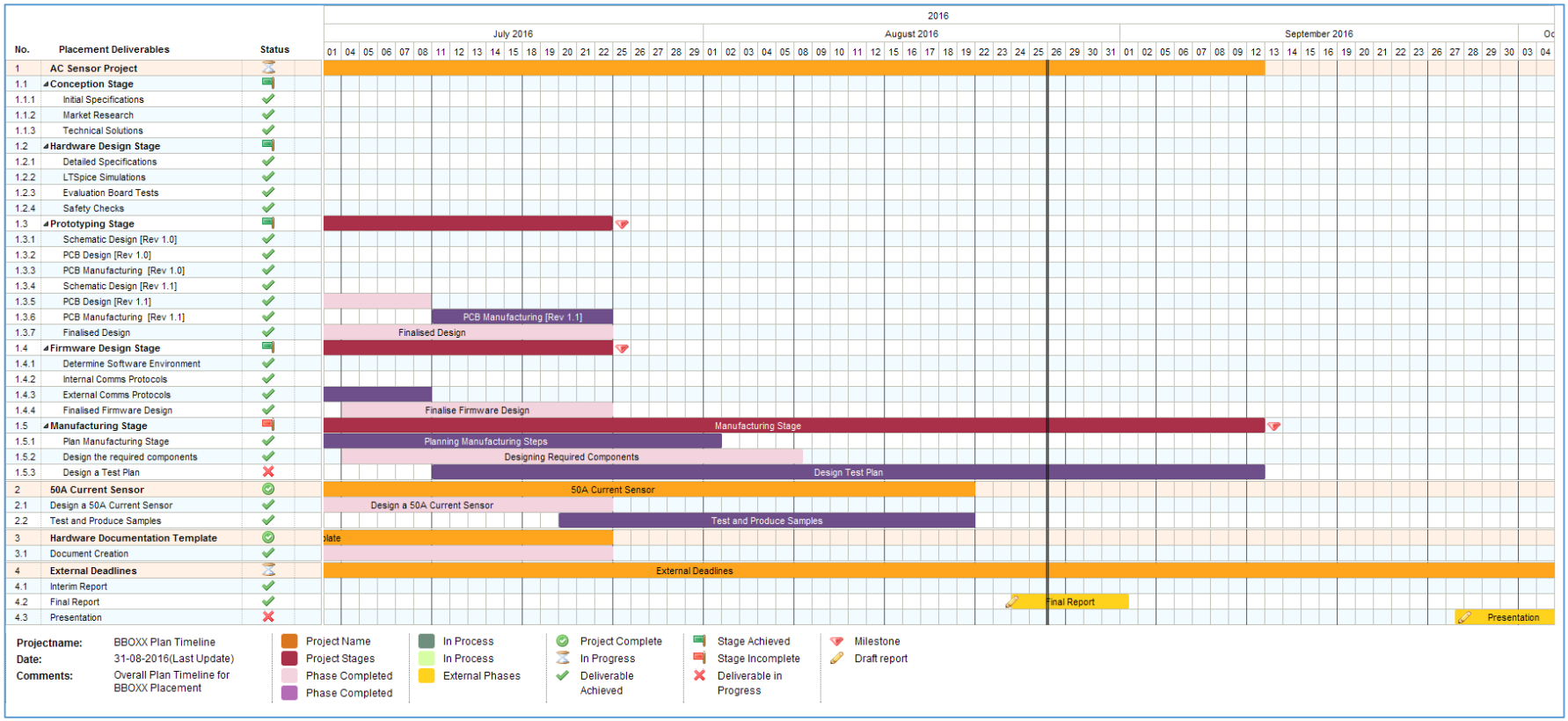

Figure 3.2: Project Schedule from July to September 2016

This Gantt Chart allowed for the planning of situations when there was time available, such as during the 2 weeks PCB manufacturing. During those times, I was working on the firmware design stage, and also researching on the manufacturing stage deliverables. Overall, this Gantt chart allowed me to efficiently use my time at BBOXX by allowing me to keep an eye on all the deliverables I had to meet, and to also allow me to work on getting all the deliverables done by the deadlines set.

From the chart above, it is clear that the deliverables for this project were all met, and that the project plan was followed for the entirety of the project.

4. Details of Work

In this section, a step-by-step process of the methodology for achieving the deliverables for each project will be given.

4.1 AC Sensor Project

4.1.1 Conceptual Stage

Phase	Deliverables	Approaches Taken
Conceptual	○ Determine if the AC Meter market is worth entering. ○ Determine the initial specifications required for the AC Meter, and decide on the best possible method for sensing the readings.	○ Market Research on current metering solutions ○ Build up initial specifications of AC Meter for design ○ Choose between different sensoring methods for AC voltage and current

Table 4.1: Conceptual Stage Deliverables and Approaches Taken

Market Research

A detailed account of the AC metering market research is shown in **Annex B**. From the research, it is noted that BBOXX has a competitive advantage in having existing GSM capabilities with web server management. In addition, the prices for the current meters are in the range of [£140 £200], which allows BBOXX to make a sizeable profit margin if they enter the market. Hence the decision was made to enter the market as it was advantageous for BBOXX.

Initial Specifications

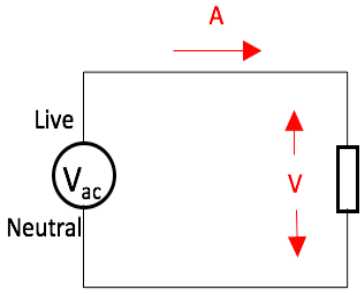

Figure 4.1a: Simplified Model of an AC system

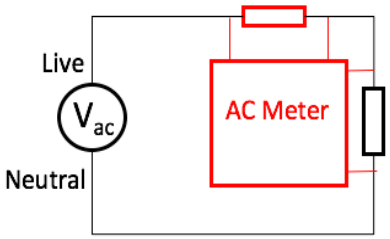

Figure4.1b: Model of an AC System with the metering system (in red) in place

An alternating current (AC) sensor is defined as an instrument capable of real-time measurements of voltage, current, power and energy values input from an AC source to a load. This sensor must also have the means of transmitting this data for the user to analyse via GSM for off-grid locations. There must also exist the means for the user to remotely disconnect the load from the mains.

Figure 4.1a above demonstrates a simple AC voltage system. The live current flows from the mains into the load. In order to measure the current and voltage applied on the load, a current sensing resistor must be introduced into the circuit as demonstrated in **Figure 4.1b** with the AC metered circuit in red. By doing so, the AC meter can measure the voltage falling across the current resistor and obtain a value for the current. This sensor will interface with the BBOXX Hub which already has existing GSM communications and power supply management capabilities.

Choice of Sensoring Method

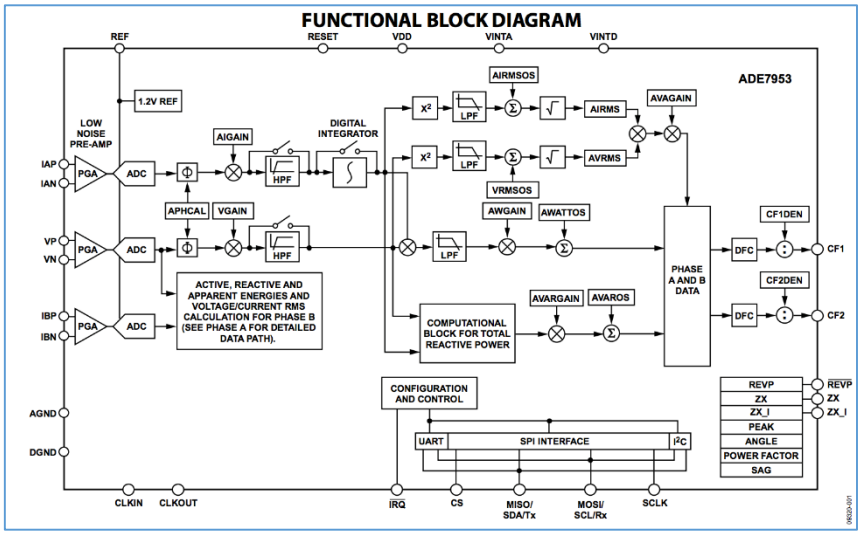

Figure 4.2: Block Diagram of the ADE7953- Chosen choice for sensing voltage and current

The choice of sensing the AC method was compared between 4 different choices and is covered in greater detail in **Annex C**. The final choice was eventually to use an application specific integrated circuit (ASIC), which was the **ADE7953**. This ASIC had the advantage of being an accurate, and low cost sensing device which was also capable of DC measurements. However, one drawback was that a large amount of time was required to understand the working of this ASIC. This came from looking through the datasheet[2] and understanding how the ASIC worked. This was important for coding the firmware behind the SPI interface required, and also for configuring the voltage and current input values such that they were calibrated correctly.

[2] http://www.analog.com/media/en/technical-documentation/data-sheets/ADE7953.pdf

4.1.2 Hardware Design Stage

Hardware Design	○ **Determine the circuit required for the AC Meter.**	○ **Simulate circuits on LTSpice** ○ **Safety Measures** ○ **Detailed Specifications**
	○ Test and determine the components required for the AC Meter circuit	○ Obtain Evaluation boards and test separate components

Circuit Tests

SPICE Simulations were used for determining the appropriate circuit for the AC Meter. This allowed experimentation with higher voltages without the risk of danger, and also for finding the component values required. This step was important in drafting the eventual schematic for the circuit.

Due to the high voltages and currents involved, a test system was conceived for the high voltage inputs in order to prevent any harm to the user and the equipment. This was done by applying isolation to the circuits involved, and using DC-DC converters with opto-couplers during the design of the circuit.

After the simulations were done, the specifications were further modified to include the safety measures taken and the changes to the components. **Figure 4.3** shows the overall block diagram of the meter for design, with the various inputs and outputs of the system.

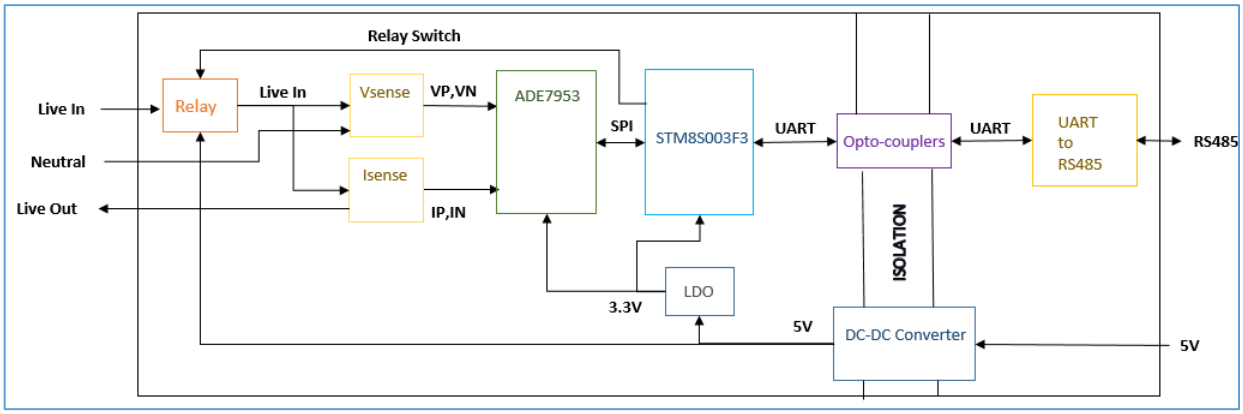

Figure 4.3: Overall Block Diagram of AC Sensor Device

The different blocks in the system represent the different modules in the system, with each having a required role and purpose in the function of the AC Sensor. The Relay Module represented the switching required for connecting and disconnecting the load. The Vsense module represented the potential divider required to step down the voltage to an appropriate level for the ASIC, while the Isense module represented the current sensor circuit required for a value of the current to be read. The ADE7953 module represented the ASIC with all its peripherals (clock, power supply), while the STM8003F3 module represented the microcontroller for controlling all communication protocols on the sensor. The LDO module represented the low-dropout regulator used for stepping the 5V power supply value to a 3.3V value for the ICs, and the Opto-coupler module represented the optocouplers required for isolating the signals from the live side to the signal side. Similarly, the DC-DC Converter isolated the 2 sides from each other, while last but not least, the UART to RS485 IC converted all the UART signals from the microprocessor to a RS485 signal and vice versa for receiving and transmitting communications to a RS485 Bus. This block diagram represents the entire hardware design of the AC Sensor.

Evaluation Board Tests

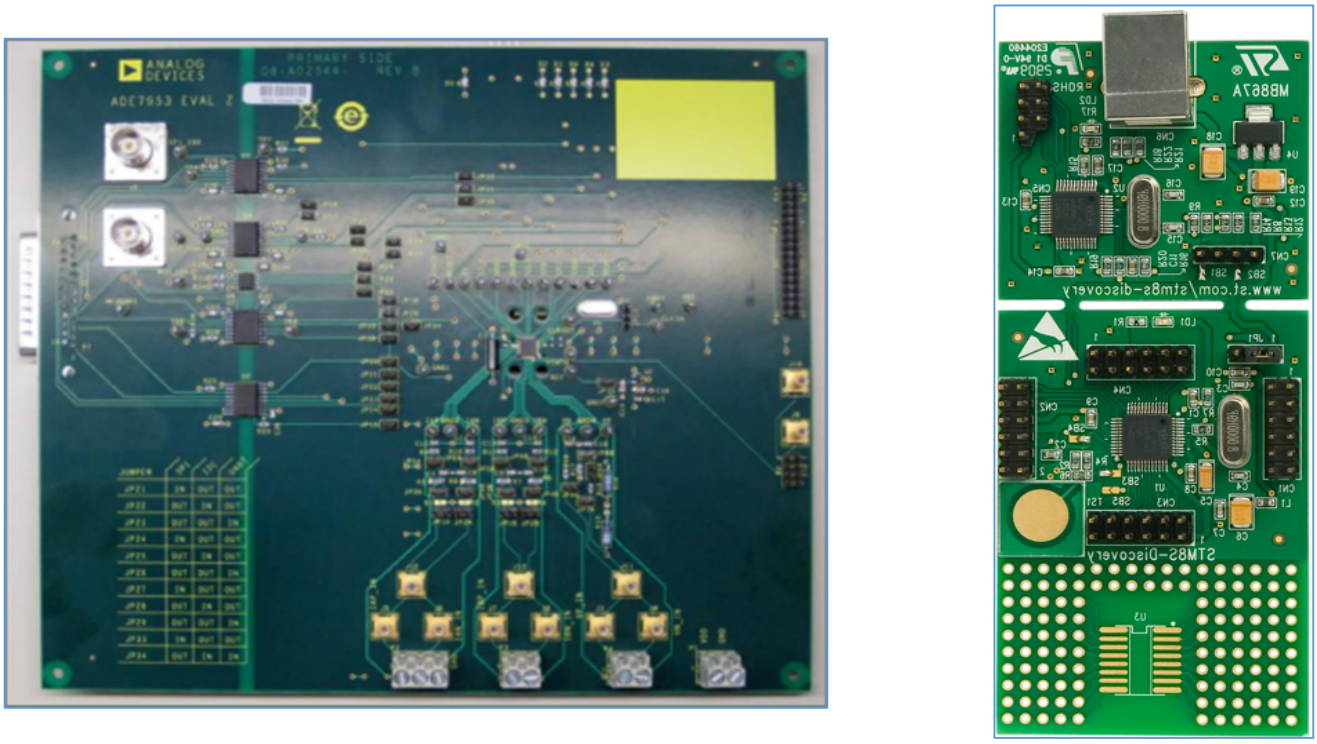

Figure 4.4: Evaluation Boards used in Design evaluation: EVAL- ADE7953(left), STM8-DISCOVERY(right)

The decision was made to purchase evaluation boards for the 2 main ICs required in the AC Meter: The sensing ASIC (**ADE7953**), and the microcontroller. This was to ensure that the component chosen could work as expected, and to also see whether it was suitable for the operation required of it.

With the evaluation board, it was straightforward to test the **ADE7953** and **STM8003F3** devices, since they came with proprietary software installed. For the **ADE7953**, it was an opportunity to test the accuracy of the device for the readings required (Voltage, Current, Power and Energy). The **STM8003F3** software was unusual in the fact that the compiler was not freeware as advertised by STMicroelectronics. This added an unnecessary cost to the device, but was overcome by finding an open-source compiler. This will be covered in greater detail in **Section 4.1.4**.

The evaluation boards were important as they allowed me to test the key components I needed, without compromising on my safety. As can be seen on the **ADE7953** evaluation board, there was isolation on the inputs to the microcontroller, which allowed for safe testing. The STM8 discovery board also contained an easy interface to the PC via a USB link, which allowed for easy flashing and programming.

The first tests I conducted involved using the **STM8** evaluation board to communicate to the **ADE7953** evaluation board via the SPI interface. Once that worked, I used the **STM8** board to communicate via the UART interface to the PC. Having these 2 boards greatly assisted me in the initial firmware design stages for the AC Sensor.

4.1.3 Prototyping Stage

Prototyping	○ Design the Schematic required for AC Meter.	○ Gain proficiency in Altium Designer ○ Split schematic into different modules for ease of design
	○ Design the PCB required for the AC Meter.	○ Understand how PCB layouts are done ○ Finalise PCB layout
	○ Finalise PCB Design	○ Rigorous Testing on PCB done ○ Redesign PCB for improved revisions

Schematic Design

Altium Designer Proficiency

The first task required was to gain proficiency in Altium, which was the tool used by BBOXX for schematic and PCB design. Altium is a highly customizable software with the tools to vary almost all parameters, which also meant that to utilize the software, it was important to learn most of these tools as much as possible. This took about a week of learning through online manuals, and to also experiment with different designs myself. However, the most important knowledge was from the experienced employees in BBOXX, where they explained the steps and methodologies required for producing schematics and designing PCB layouts.

Final Schematic

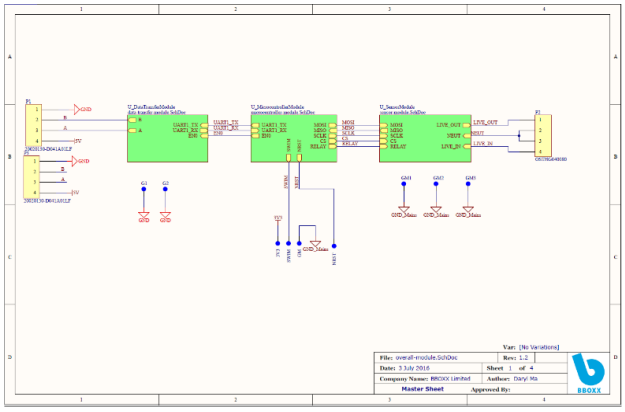

Figure 4.8: Master Sheet

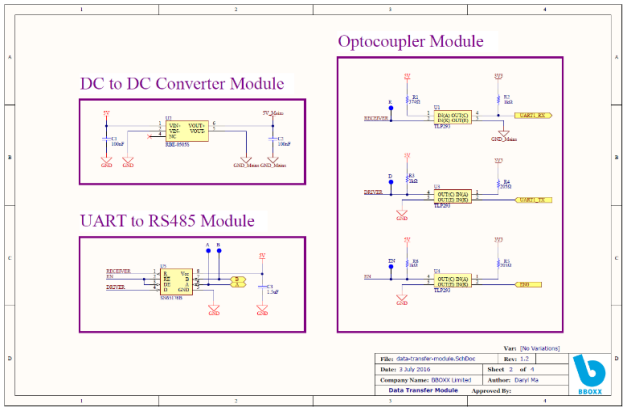

Figure 4.8: Data Transfer Module

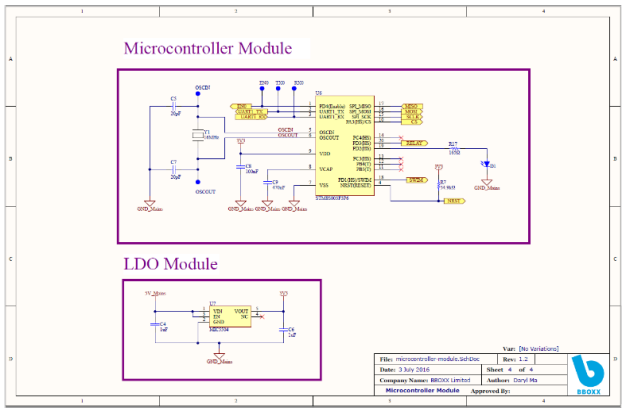

Figure 4.8: Microcontroller Module

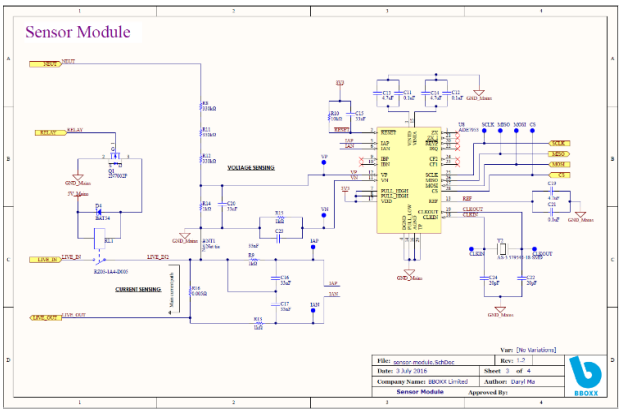

Figure 4.8: Sensor Module

The overall schematic in Altium is split into 4 separate sheets, with the master sheet indicated in **Figure 4.5**. **Figure 4.6 – 4.8** display the 3 schematics representing the data transfer module, microcontroller module and sensor module respectively. The schematic design for this sensor was thought out over time, with constant updates due to the rigorous testing done over the entire prototyping stage.

The first step in designing the schematic was to determine the components required. There were a number of factors that contributed to choosing the component. First, the requirements for each component was chosen, and the cost compared for components that met those requirements. The requirements for the components could vary according to each other. For example, the DC-DC converter had to have a large enough pitch (distance between the pins) such that it successfully negotiated the isolation region. For the LDO module, it was essential that the input voltage to the LDO could vary within the range of 4.5V to 5.5V due to the output of the chosen DC-DC Converter fluctuating within that range. Sometimes, it could simply be that the height of the component could not meet the clearance required, and thus it would fail the requirements.

After choosing the components, it was a case of wiring up the schematic as per the simulations, and also applying some analogue electronics. There were times when the only way to determine the correct schematic was by simply trying out the circuit in LTSpice and determining if it would work well.

This tended to require lots of testing before the final schematic was determined with the components chosen. For a greater detail of the different components chosen and the choices made for the schematic, please refer to **Annex B**.

Printed Circuit Board(PCB) Layout

The schematic was exported onto the PCB designer, where the components were placed. The components each had an individual footprint that had to be designed specifically, which took some time to get right. Once that was done, the components were laid out in a manner that minimised the PCB track usage. This reduced the possibility of noise signals entering the circuit, especially for the case of the inputs into the ASIC. For a greater detail into the design of the PCB board, please refer to **Annex E**.

The first prototype of the printed circuit board (PCB) was done, and manufactured on the 1st of June. The aim of this prototype was to conduct tests on the efficacy of the layout, and to ensure that this could efficiently and accurately measure voltage and current values. **Figure 4.1** displays the PCB layout of the board.

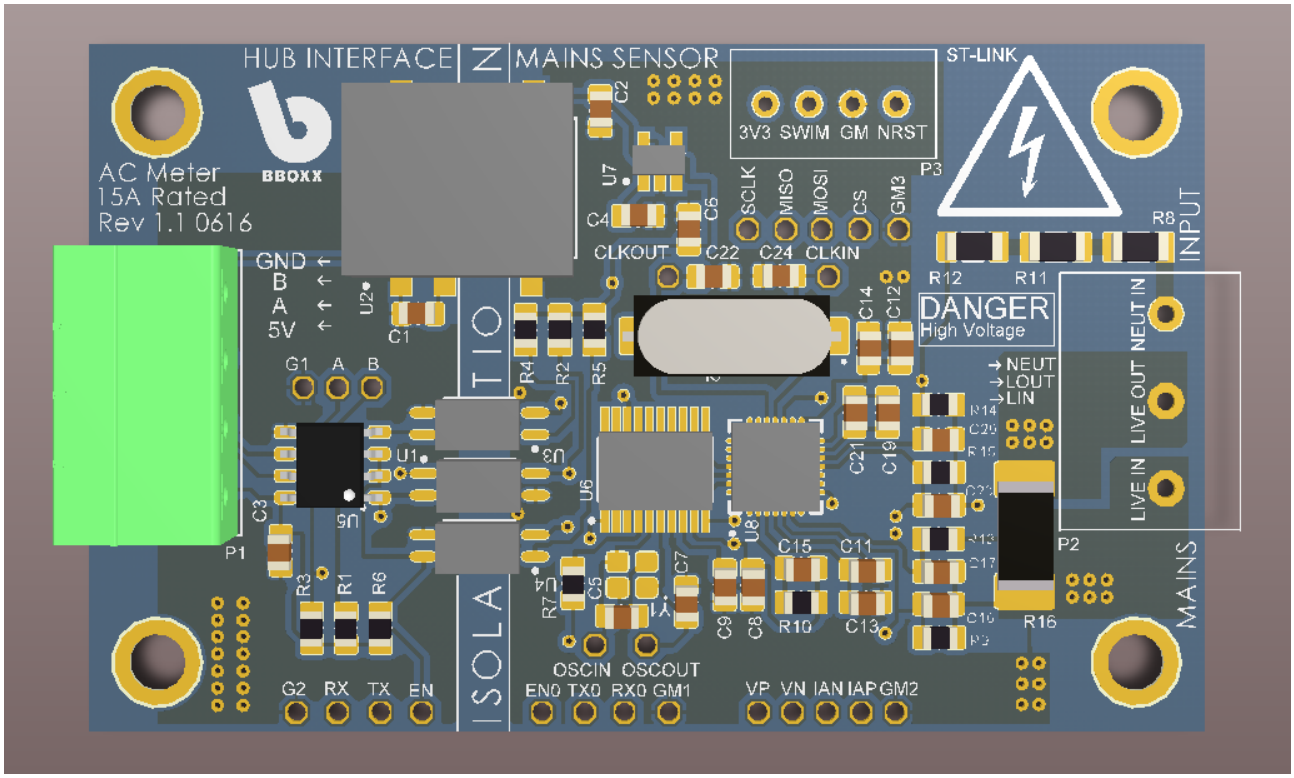

Figure 4.9: AC Meter Prototype 1

The tests revealed the flaws in the initial design. The testpoint vias in prototype 1 were too small, and the DC-DC Converter did not function as expected. In order to correct this, the schematic was corrected and the components were changed.

Redesigned PCB for improved operation

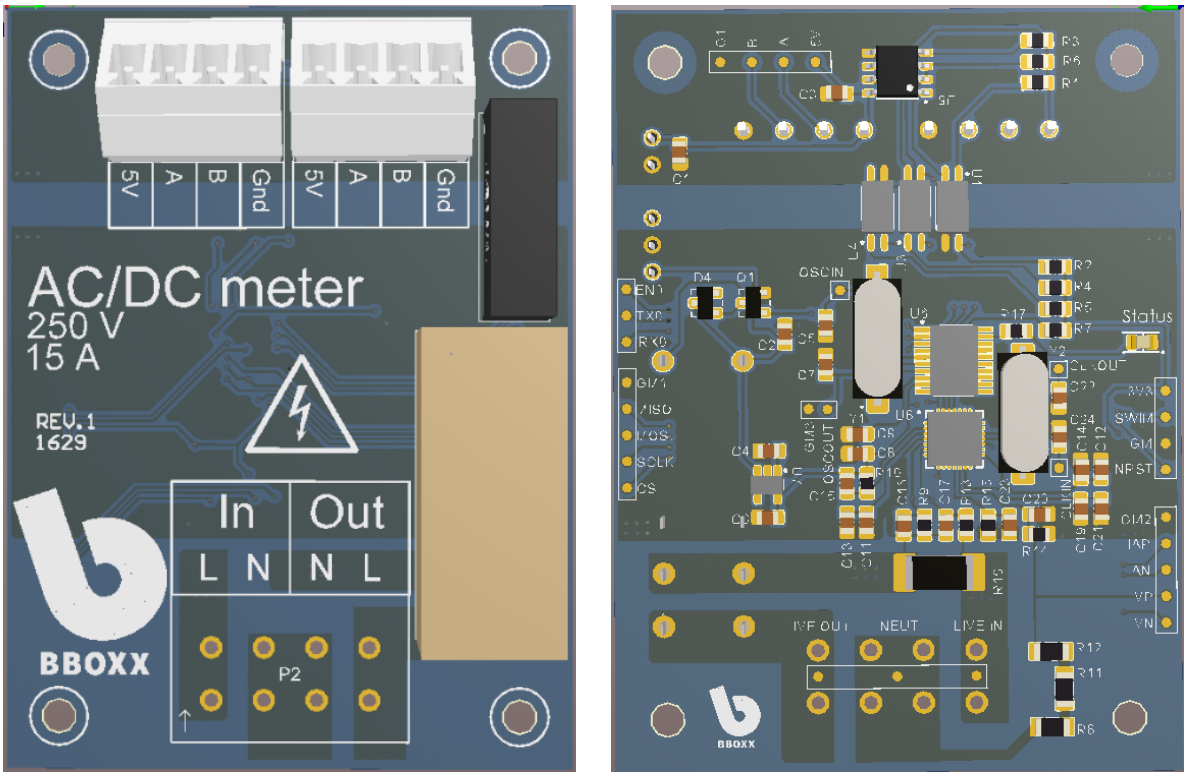

Figure 4.10: PCB Prototype 2

Figure 4.10 displays the designed PCB prototype 2. Notable differences include the board size, which was modified to be able to fit DIN rail enclosures (72mm), for through-hole mounted components to be placed on top, and lastly for the test points to be modified from vias to pads, which was to allow for easier flashing and programming. This will be explained in greater detail in **Section 5.**

Another change was connectors, which were now placed at a 45° angle. Lastly, the relay is added to the final design.

Figure 4.11 displays the prototype 2 sensors after soldering and mounting all components. The sensor worked as expected, and this was decided to be the final design for the sensor as it satisfied all requirements. It was also robust, and worked well during the testing.

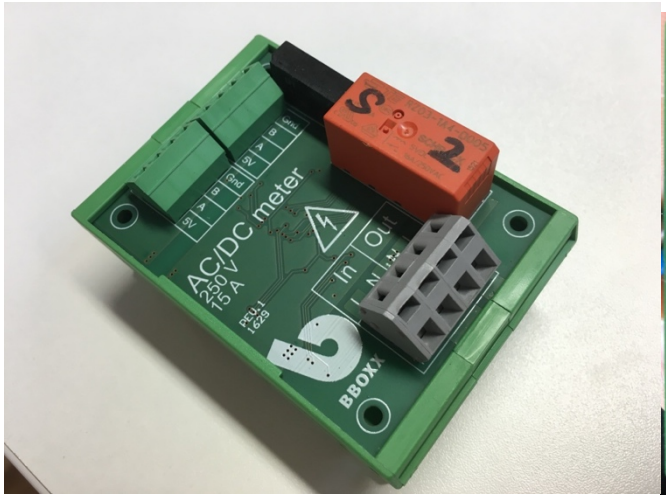

Figure 4.12 Top View of AC Meter with DIN Rail Enclosure

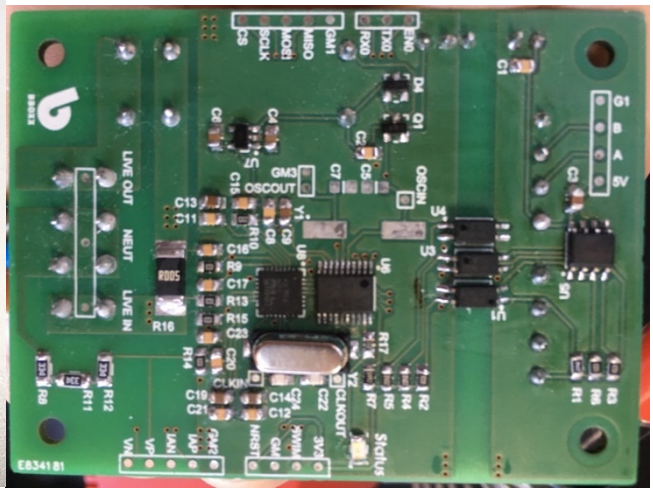

Figure 4.12: Bottom View of AC Sensor

4.1.4 Firmware Design

Firmware Design	**Firmware Overview Plan**	**Determine the Software Required** **Plan for how the modules in the system communicate**
	Coding individual Modules	Determine the functional Requirements Begin working on individual Modules Put the code together

Software Requirements

The proposed software development tools list provided by STMicroelectronics is Raisonance Software[3] and Cosmic Software[4]. Upon further examination, it was ascertained that these products were only available for trial periods, and not freeware as advertised. This would add on unnecessary costs to the component, and hence the decision was taken to find an alternative compiler. The chosen compiler was the **Small Device C Compiler (SDCC)**, an open source program that could target STM8 chips.

The remainder of the software was mainly a text editor to edit code, for which **Sublime Text** was used, a software for receiving signals from the sensor via the RS485 Bus, for which I used another open source software, **PuTTY**, which is a terminal emulator that translated signals received from a COM port.

Another challenge faced was the flash size of the **STM8** microcontroller. After compiling the **STM8** peripheral libraries, this left around 2kbytes of space left for additional code. This meant that the peripheral libraries had to be edited in order to include only the libraries required.

[3] http://www.raisonance.com/stm8-compiler.html
[4] http://cosmic-software.com/

Firmware Plan

In order to begin with the firmware, it was important to understand what was required for the sensor to communicate with the **BBOXX Hub. Figure 4.11** describes the Hub and Sensor interface. The Hub would be the master on the RS485 bus, with all the sensors listening in to any commands. Hence, in order to correctly send a command to the desired sensor, a method needed to be derived whereby the individual sensor could decode the message sent.

Figure 4.12 displays the block diagram of the sensor, highlighting the required communications protocols to be coded in the microcontroller. The UART communications protocol would have to be used to send values of the data gathered to the UART to RS485 interface, while the SPI communications is used to gather data from the **ADE7953**. The microcontroller also has to be able to control the relay switch for turning on and off the load, and to also control the receiving or transmitting modes to the UART to RS485 IC.

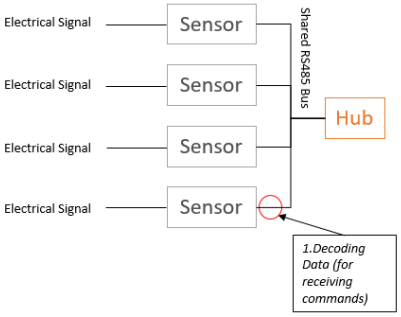

Figure 4.14: Daisy-Chaining of AC- Sensor

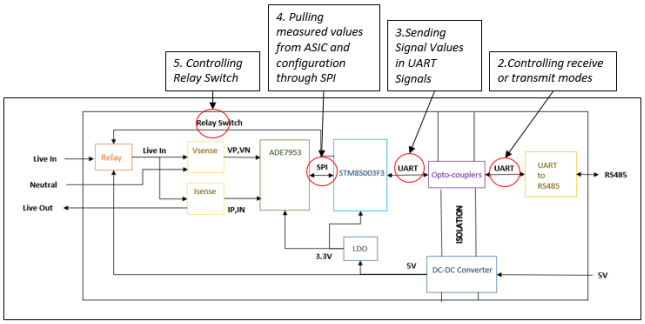

Figure 4.14: Internal Communications within AC Sensor

This initial design led to the firmware requirements, which was important in finalizing the code the AC Sensor firmware.

Firmware Requirements

Auto Addressing

The addressing for the AC Sensor is done by the Hub. This is to overcome the issue of conflicting addresses for each sensor when multiple sensors are connected to the same Hub.

The method chosen to overcome this problem is as such: On manufacturing, a unique 32-but serial number will be assigned to each sensor.

Processing Hub Requests

The AC Sensor must be able to decode any messages received from the hub, understand the command required, and reply with the data wanted.

Acquiring Data

The values required by the hub must be calibrated and calculated in the microcontroller.

Sending Data to Hub

Data sent to the hub must be compressed into a standard protocol.

Error Checking

Checksums will be implemented for cases when either the data loss occurs, or when collisions occur. This will reduce the probability of errors in transmission.

Modularized Code

The code for the firmware was split into basic functioning blocks. This was done in order to modularize the code and thus minimize the amount of firmware required due to the code limit of 8kbytes. By splitting it up into modular functions, the rewritten stretches of code was reduced and thus space was saved.

The code was structured into the following header and source files:

1. `ade7953_interface.c`
2. `rs485_interface.c`
3. `timer.c`
4. `stm8_specific.c`
5. `command.c`
6. `main.c`

ADE7953 Interface

The `ade7953_interface.c` source file dealt with the SPI configuration, reading and writing to the ADE7953 from the microcontroller. In this case, the microcontroller controlled all the communications as it was the master in this configuration. This file also dealt with the DC and AC configurations for the ASIC, which was done by setting the high pass filter. Lastly, the function for obtaining the voltage, current, power and energy values from the ASIC was done in this source file.

RS485 Interface

The `rs485_interface.c` consists of the functions for initializing the UART configurations and for interfacing with the UART to RS485 IC. This function also contains the UART Read and Write function, and a function to determine if there is still available space left in the UART buffer. Lastly, this file also contains the function for the **BBUS Protocol**, a unique addressing protocol designed for interfacing between the AC Sensor and the Hub.

Example Packet Layout										
1 Byte							…			
Length	Address				CMD	Data			Checksum	

Figure 4.15: BBUS Protocol Packet Layout

Different communication protocols were considered, such as the Modbus protocol and the TCP/IP protocol. However, the chosen protocol in the end was a unique protocol.

The **BBUS protocol** consists of 5 different fields:

1. The **length** field, which indicates the number of bytes of the packet (inclusive of the length field)
2. The **address** field, which is the standard 4 byte – long serial number of the hub, or 0xFFFFFFFF for broadcasting.
3. The **command** field, which indicates the request required by the hub.

4. The **data** field, which varies depending on the request and whether additional data is required.
5. The **checksum**, which is a 2-byte long value calculated in the function to determine errors in transmission.

The **checksum** worked by inputting the bytes of all the previous bytes in the frame, and calculated a 16-bit integer based on those values.

Timer

The `timer.c` contains the configuration function for the timer in the STM8 that allows for 1ms delays. This is done by dividing the internal 16MHz clock by a prescaler value of 16000(0x3E80). Doing so allows for a clock that has 1000 ticks per second, which gives us a 1ms clock.

The remainder of the functions in this file contains a timeout function, which serves various purposes. For example, for cases when the communications from the BBOXX Hub is interrupted, and the packet stops transmitting halfway through, the timeout allows the UART receive to reset.

Lastly, the delay function which is important in waiting for peripherals to start up. Sometimes the code runs faster than the peripherals, which is why delays need to be implemented to prevent the code from overrunning them. A lot of time was spent determining the peripherals that required delays, as this can occur in unexpected places. This can be seen in the code in **Annex F.**

STM8 Specific

The `stm8_specific.c` file contains the microcontroller functions required for the running of the chip. This includes the relay configuration, the checksum function, the EEPROM write function, the LED configuration and the random number generator function.

The relay and LED configuration are similar to the SPI and UART configurations, for which the required pins are configured as push-pull outputs, before being set to logic high or logic low states.

The EEPROM write function works by first assigning a pointer to the address where the EEPROM registers are held. A key has to be entered into the FLASH_DUKR register in order to unlock the EEPROM, before data is written into the EEPROM, and locked.

The random number generator used is a linear congruential generator, which was compared to a linear feedback shift register, both random number generators. The function returns a value between 0 and 310, which is useful for the broadcast function to be mentioned later.

Commands

The `command.c` file consists of the possible commands sent from the **BBOXX Hub**, and utilizes functions from all the other files. The commands could be separated into manufacturing commands, and runtime commands.

Manufacturing Commands

These commands are performed by the sensor on startup for storing values in the EEPROM that will remain even if the sensor is turned off.

Calibration

For calibration of the sensor to output accurate voltage and current readings, it was essential to calculate the offset voltage and current value – The voltage and current values read on the sensor when no input is given.

Another point to note was that the voltage and current values read were given in LSB values as read by the ADC in the chip. It was important to scale the values to the actual input values.

The PC sends the command with 2 data values indicating the current DC test voltage and current currently applied to the ASIC. This is done during the manufacturing phase, where the scaling factor converts the LSB values to mV values. By inserting test values into the circuit(i.e 10V and 3A), and sending these values to the sensor, the microcontroller calculates the scaling factor with the following equation:

$$Scaling\ Factor(LSB/mV) = \frac{Voltage\ Readings\ from\ ASIC(LSBs) - Offset\ Voltage(LSBs)}{Test\ Voltage(mV)}$$

To calculate the actual voltage values,

$$Voltage\ Actual(mV) = \frac{Voltage\ Readings\ from\ ASIC(LSBs)}{Scaling\ Factor(LSB/mV)}$$

These two equations are used to calculate the current and voltage values read from the sensor.

Assigning Addresses

The address is assigned on manufacture by sending a 32-bit serial number to the sensor that is stored in the EEPROM. This allows for the sensor to each have a unique serial number, and prevents any possible conflicts on the RS485 Bus.

Runtime Commands

AC Readings

This command configures the ASIC for AC readings by configuring the high pass filter to be switched on. The register values are then read from the ASIC through the SPI protocol, before conversion from LSB to mV,mA,mW and mJ values is done in the microcontroller. These values are then sent via the UART protocol to the UART to RS485 converter.

DC Readings

This command configures the ASIC for DC readings by configuring the high pass filter to be switched off. The reading process is then similar to the AC readings method.

Relay Switching

The relay can be switched on or off by setting the microcontroller pin to a logic high or low. The state of the relay is also stored in the EEPROM such that it remembers the last configured state in the event of the loss of power.

Broadcast

The broadcast occurs when the BBOXX Hub has to detect the addresses of the sensors attached to the RS485 Bus. This command had to be designed such that only one sensor was transmitting at a single moment. The method decided upon is shown in **Figure 4.16** below.

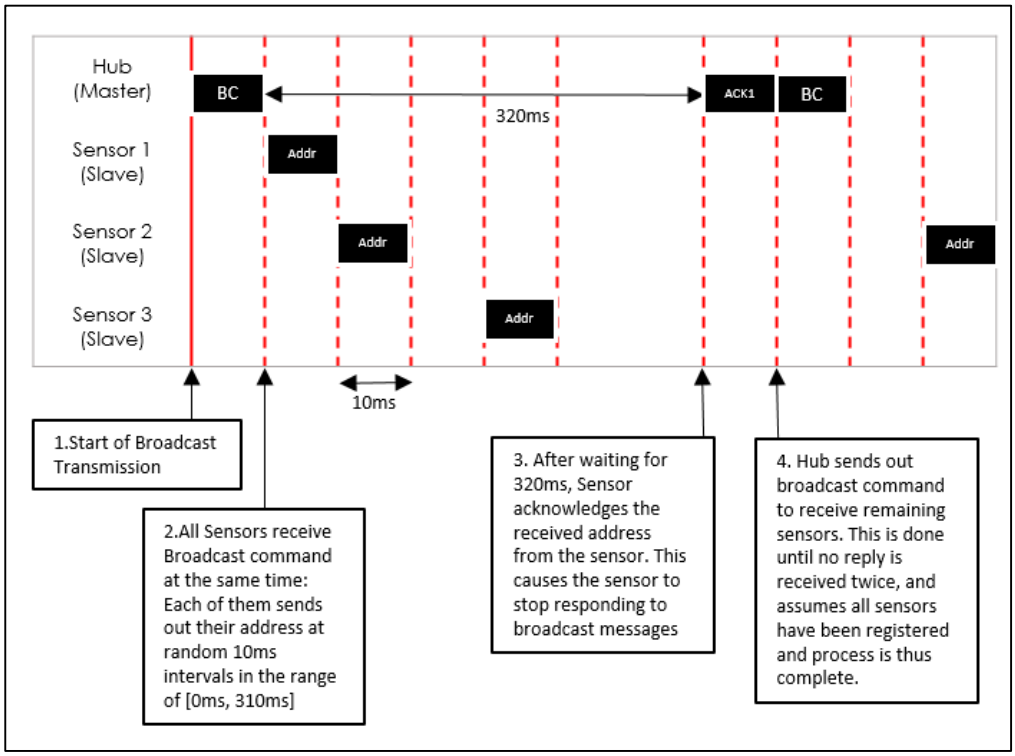

Figure 4.16: Broadcast Command Design

As the Hub sends a broadcast command, each of the sensors sends their addresses at random 10ms intervals in the range of [0ms, 310ms]. The 10ms interval was based on the broadcast command taking around 9ms to send[5]. Assuming 32 sensors to be connected to the RS485 Bus, the interval is chosen as [0ms, 310ms]. The Hub then waits 320ms for all sensors to finish transmitting.

At the end of the 320ms phase, the Hub sends an acknowledgement with the correct address to the sensor that it has correctly received, before sending out another broadcast message. This is done until no reply is received twice, whereby it assumes all sensors have been registered, and thus the process is complete.

[5] For 9600 Baud Rate, and a 10 Byte Message: Time taken to send = $\frac{10 \times 8bits}{9600bps} = 8.33ms$

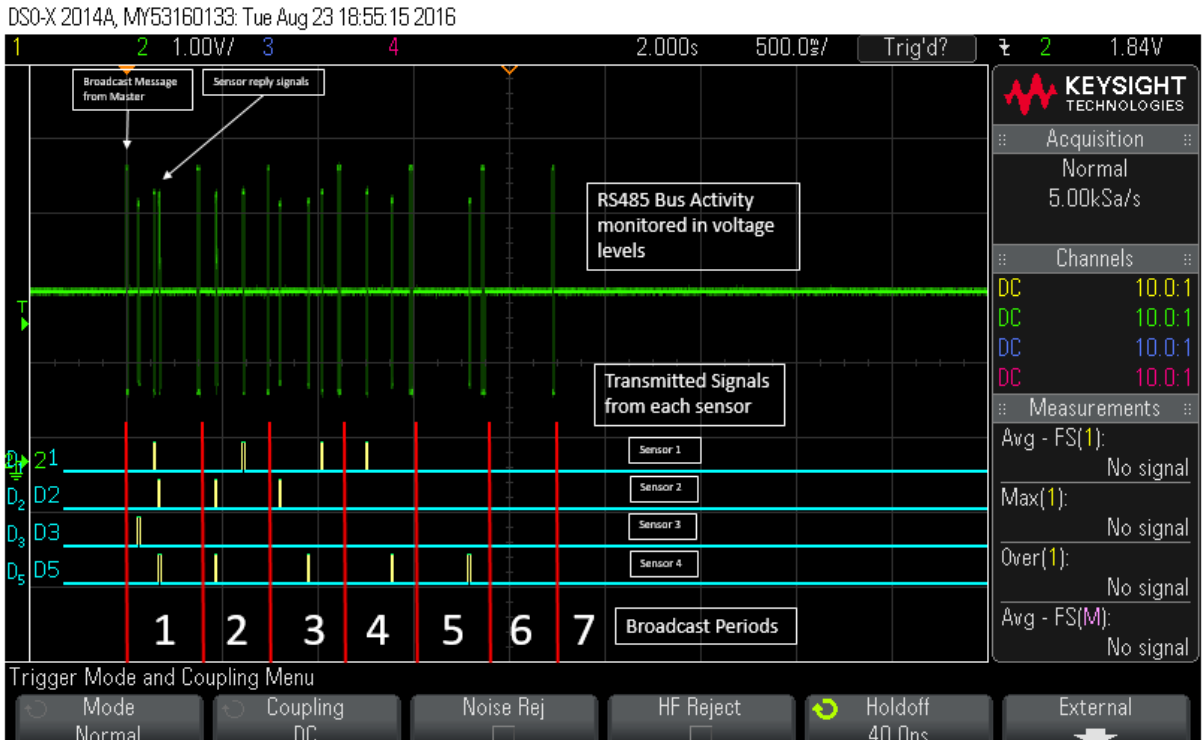

Figure 4.18: RS485 transmissions on the Oscilloscope, with the green line representing the overall voltage levels on the bus, while the D1, D2,D3 and D5 represent the transmitted signals on each sensor.

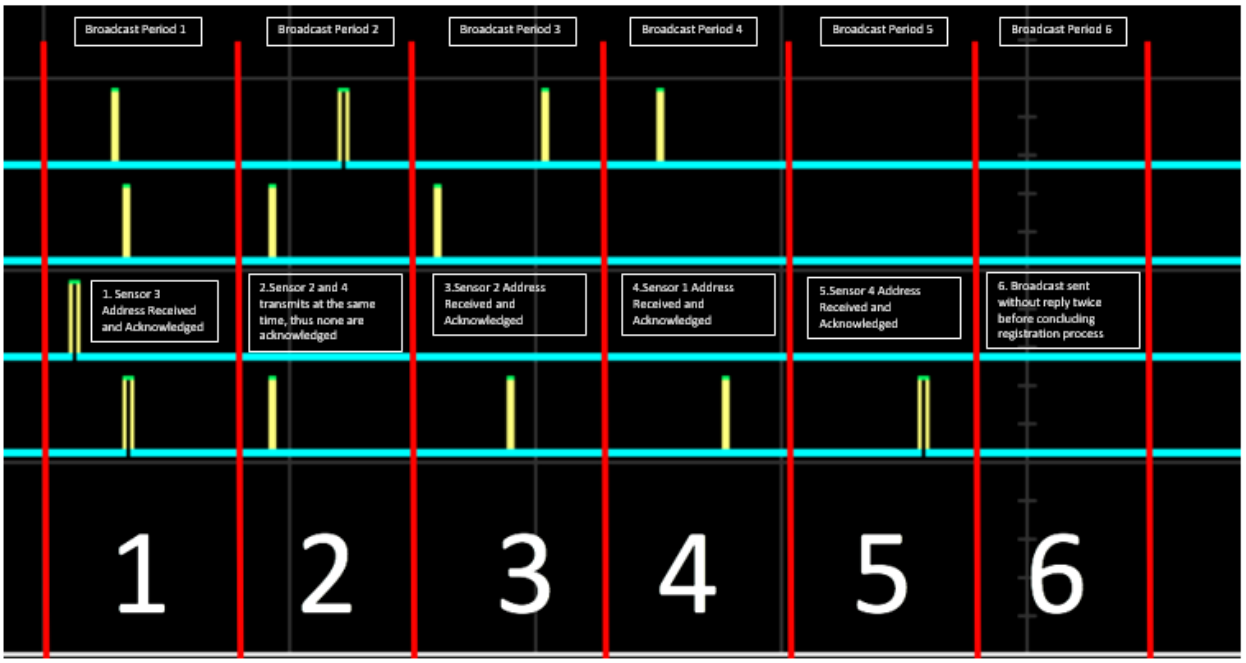

Figure 4.18: Zoomed in on each broadcast period, to explain how the broadcast function works

r

Figure 4.17 shows an actual working example of the Hub functioning with 4 different sensors for the broadcasting on the oscilloscope. As can be seen, the RS485 signals on the bus varies in voltage level, with the highest voltage arising from the BBOXX Hub request, while the sensor replies are noticeably lower in voltage. This is due to the voltage drop across the cables, due to the resistance of the cables. However, the strength of the RS485 communications is that the voltage levels do not matter, as it simply determines the difference between the A and B channels.

The broadcast command can be separated into different periods. In the **first** period, the reply from sensor 3 is received first. This is acknowledged by the BBOXX Hub. Hence, sensor 3 does not reply to subsequent broadcast commands. In the **second** period, a conflict occurs between sensor 2 and sensor 4. This causes the BBOXX Hub to detect a clash in addresses, either through the detection of a bad length or a bad checksum. Hence, it sends out the broadcast command once more. As can be seen in the **3rd, 4th** and **5th** periods, all the sensors are acknowledged one by one, before it sends out the broadcast command twice to ensure that it has received all signals.

Main

The `main.c` file consists of the interrupt function triggered when a byte is received on the UART interface. The next function, `data_readin()` then checks whether the full frame is received, and the checksum of the frame. If this is both correct, the frame is stored in an array for decoding.

In order to decode data, there had to be a way to differentiate the commands for different sensors. This was done by assigning a serial number to each of the sensors on manufacture. As mentioned previously, this is stored in the EEPROM as well as externally using a barcode sticker. The RS485 bus allows only for a single device to transmit at a single time. This gave rise to the issue of possible errors being introduced due to conflicts. The BBUSS Protocol mentioned previously reduced the possibility of conflicts.

The logic for decoding the data is shown by the UML chart on the right. The sensor waits for a received UART signal, which triggers an interrupt that stores the **byte received** into a buffer. Since the first byte is the length field, the next check would be whether the full frame is received. This is done by checking whether the length of the frame received is equal to the value stated in the length field.

This begins the **decoding process** which first checks if the checksum is correct. Hence, for situations when two or more sensors send at the same time, the checksum for the signals received would be incorrect.

Lastly, the address is determined for the sensor. If the address is correct, or if it is a broadcast address(0xFFFFFFFF), the sensor performs the required commands.

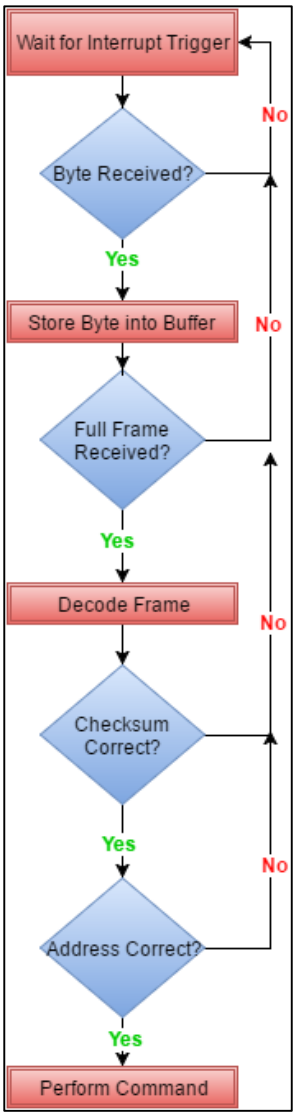

Figure 4.19: Control Flow for Decoding Data

4.1.5 Manufacturing Stage

The manufacturing stage involved the planning and process management of the device during the manufacturing of the device in the factory. As stated previously, calibration and address assigning had to be done on manufacturing. In order to facilitate the process, the PCB was designed with test pads for programming with pogo pins, which speeds up the process for flashing.

Design Manufacturing Process

Figure 4.20 displays the setup for the manufacturing phase of the sensor.

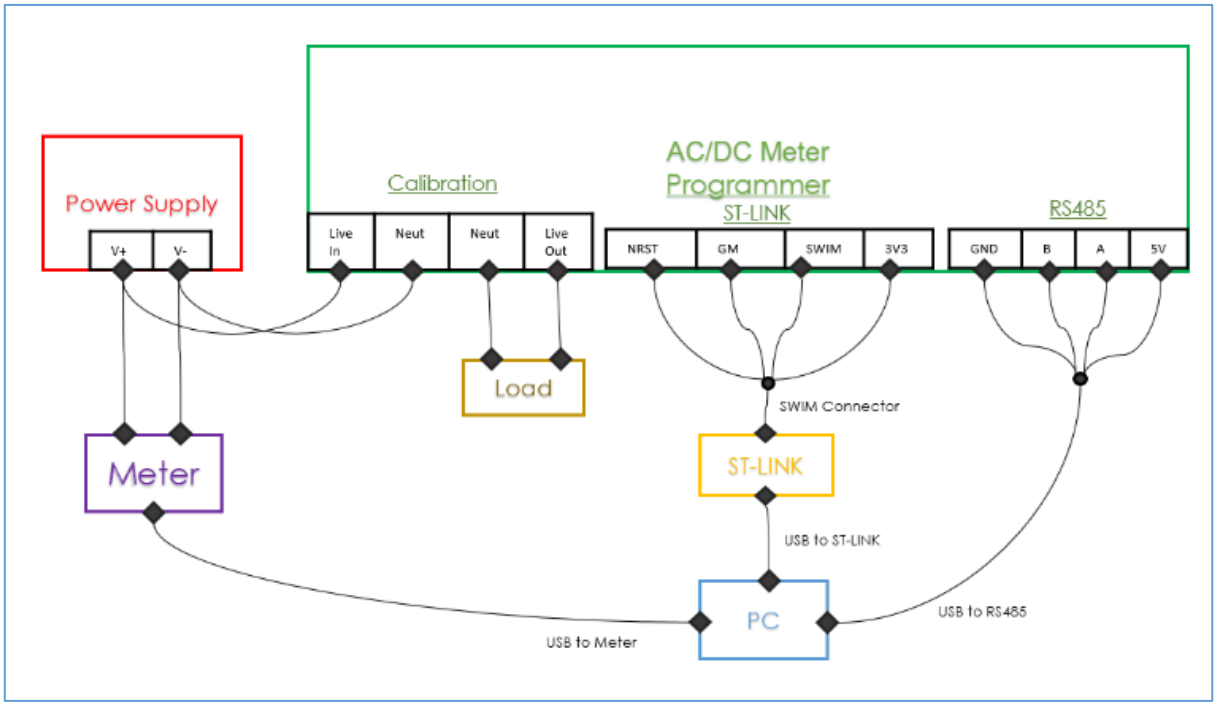

Figure 4.20: Manufacturing Setup

The PCB was designed for easy programming in mind. However, with that, it was decided that it could also be calibrated and assigned a serial number at the same time. The steps to do so are shown in the setup above.

Steps:

1. Flash the microcontroller through the ST-LINK cable

2. Calibrate the Offset: With the power supply switched off, apply the DC offset calibrate function

3. Calibrate the Sensor: The power supply is switched on, and the relay is connected, which links the load to the power supply and hence a current is drawn. The meter than outputs a voltage and current value read to the PC. The PC obtains these values, and inputs them into the function for calibrating the meter. The LSB/mV value is then stored in the EEPROM of the microcontroller.

4. The serial number of the sensor is then assigned to the device, before the barcode sticker is placed.

5. If failure occurs at any point during these steps, the sensor is removed and checked for quality purposes.

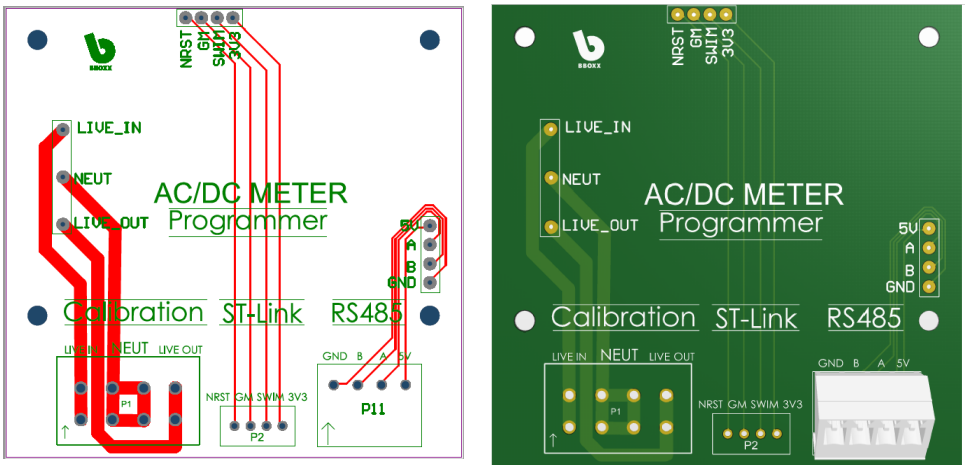

Figure 4.22: Programmer PCB design with tracks (left), 3D model of Programmer (right)

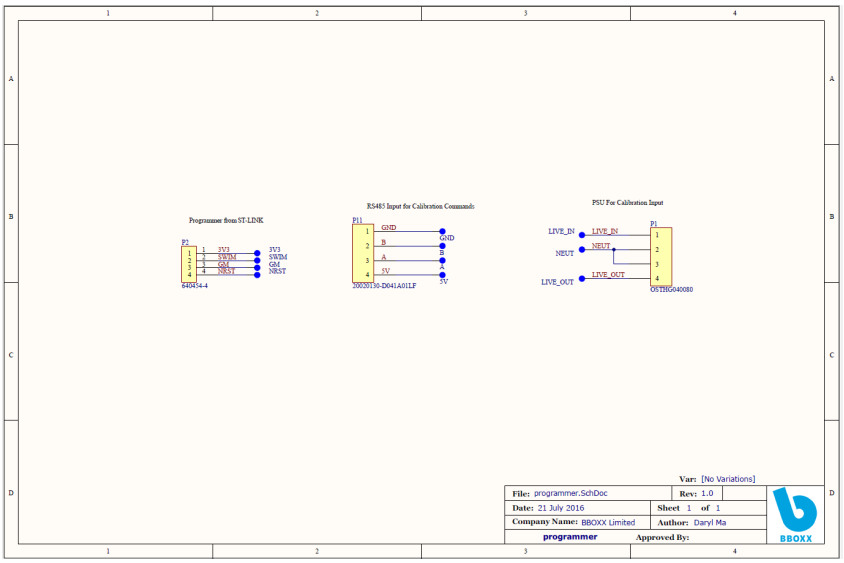

Figure 4.22: Schematic of the Programmer PCB

Figure 4.21 displays the PCB design of the programmer for programming the AC Sensor, with the schematic in **Figure 4.22**. The schematic design for this circuit was relatively easy, as it consisted of the 3 connectors, 1 for the voltage input, 1 for the programming input pins and 1 for the RS485 bus connector. The trickier part was placing the pogo pins to correspond with the test pads on the AC Sensor.

After the PCB was manufactured, the components were soldered on and tested with the AC Sensor for flashing and calibration. **Figure 4.23** displays the set-up for the entire process, with **Figure 4.24** showing the process when an AC Sensor is getting flashed. The setup worked as expected, and this is the manufacturing process plan that will be used during the first process in the factory.

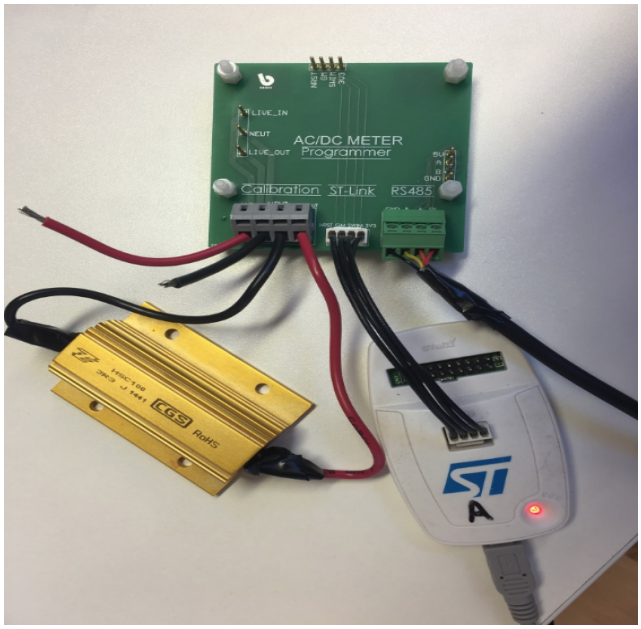

Figure 4.24: AC Sensor programmer setup, with the load and programmer. The power supply is disconnected.

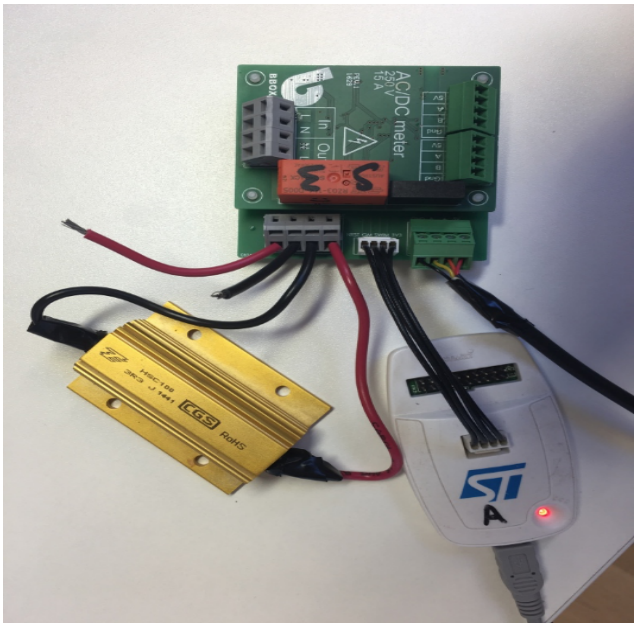

Figure 4.24: AC Sensor programmer with a sensor being flashed

Future Work

For each of the 3 phases(Hardware Design, Firmware Design and Manufacturing), there are possible optimisation features that could be implemented to improve the AC Sensor. For the last few weeks of the placement, I will be working on the following: For the hardware, a possible cost optimisation feature could be to change the RS485 connectors used to RJ45(i.e Ethernet) connector. The reason this is done is due to the larger usage of Ethernet cables, which would further lower the cost of the sensor. This would have to be done in tandem with the BBOXX Hub, and hence this would be something that is dependent on the requirements of the BBOXX Hub as well.

For the firmware, I am attempting to use the sleep states on the microcontroller to lower the overall power consumption of the AC sensor. The power consumption of each sensor limits the number of sensors capable of being run on the RS485 bus, since the power for all the sensors is attained from the BBOXX Hub.

Lastly, for the manufacturing phase, it is important to run test software that determines whether the sensors are faulty or not. At the same time, it would be good to automate the entire manufacturing process into a script. An initial development done in TKinter for the graphical user interface in python as shown in **Figure 4.25**. This is not the completed version and is still in development.

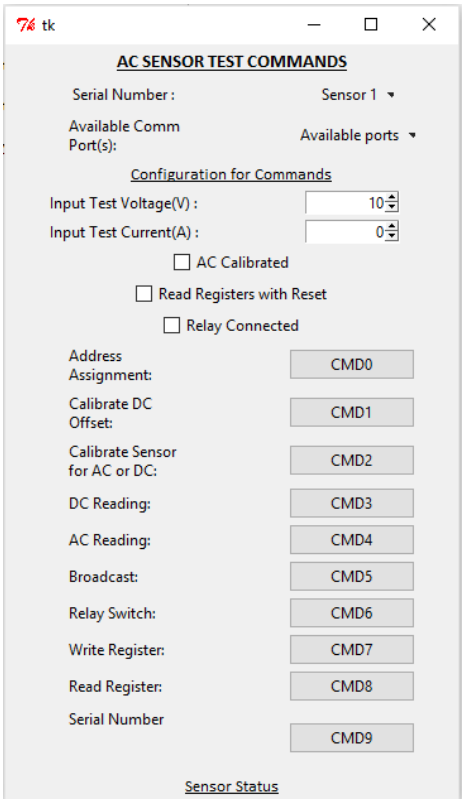

Figure 4.25: Development of Test GUI for the AC Sensor

4.2 50A Current Sensor

Deliverables	Approaches Taken
Design a 50A Current Sensor	(1) Look for components and test them
Test and Produce Samples	(2) Produce the samples and ensure they undergo proper tests

4.2.1 Designing a 50A Current Sensor

This sensor was designed for cases when a higher current rating was required. This could occur for situations when DC systems require a large current, and was the case when one of BBOXX customers requested for a current sensor that was rated for 100A, and could measure up to 50A accurately.

This meant that the AC sensor that was used had to modified due to the current rating being unable to meet the requirements. Furthermore, the cables that were meant to be measured were of 16mm in diameter, which made it unsuitable for the connectors on the AC Sensor.

The choice was eventually made to source for a non-invasive hall sensor that measured the current flowing through the device. This sensor works by producing an output voltage proportional to the magnetic field generated by the current flowing through the device.

Figure 4.26 displays the PCB designed for the current sensor. This PCB was unique since the cables had to pass through the PCB, which resulted in the shape as shown in **Figure 4.27**. The schematic for this design was straightforward, with only a low-dropout regulator, a terminal block and a few capacitors required.

Figure 4.28: 50A Current Sensor Bottom View

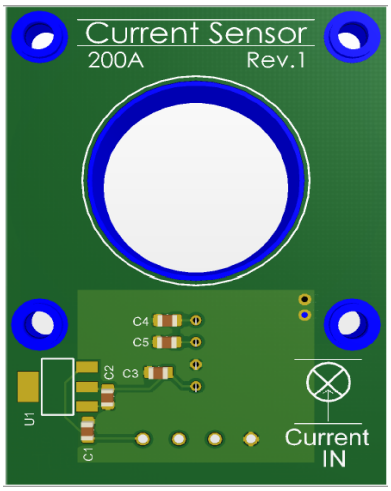

Figure 4.28:50A Current Sensor Top View

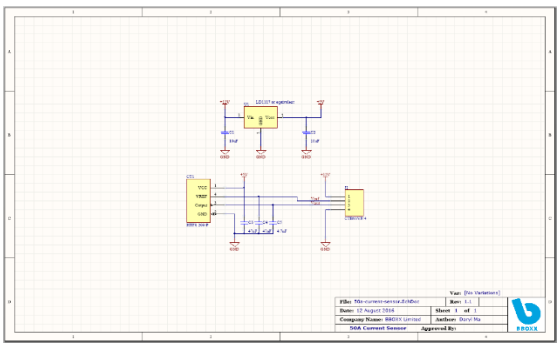

Figure 4.28: 50A Current Sensor Schematic

4.2.2 Samples and Test

Figure 4.29 displays the top view of the completed sensor, while **Figure 4.30** shows the view from the bottom. The samples were tested to ensure that they were accurate, and they functioned as expected.

Figure 4.30: Top View of Completed 50A Sensor

Figure 4.30: Bottom View of 50A Sensor

This sensor would provide a useful alternative to situations when customers require a higher current rating, or when they would prefer methods that would not result in the live wire having to be cut before insertion. This provides a non-invasive alternative to the current measuring solution, and also allows for a higher current rating and larger cables in the process.

However, the main disadvantage of this sensor is the price, with the cost of the current sensor being $16 USD, which is much higher than when a shunt resistor is used. Hence, this sensor would be considered a bespoke item, only made when customers demand it.

4.3 Hardware Template

5. Reflections

5.1 Challenges

This placement was a whole new experience for me, from the beginning of the

5.2 Personal and Professional Lessons Learnt

5.3 Future Work

5.4 Relevance to Degree Course