COST EFFECTIVE PORTABLE DIGITALLY CONTROLLED BENCH POWER SUPPLY

by:

C.G. Smith 24970875

Project Proposal

submitted in pursuit of the degree

BACHELOR OF ENGINEERING

In

COMPUTER AND ELECTRONIC ENGINEERING

North-West -University Potchefstroom Campus

Supervisor: Dr H. Marais

Potchefstroom

October 2017

It all starts here

NORTH-WEST UNIVERSITY
YUNIBESITI YA BOKONE-BOPHIRIMA
NOORDWES-UNIVERSITEIT
POTCHEFSTROOM CAMPUS

Declaration

I, **Cordre Smith**, declare that this report is a presentation of my own original work.

Whenever contributions of others are involved, every effort was made to indicate this clearly, with due reference to the literature.

No part of this work has been submitted in the past or is being submitted, for a degree or examination at any other university or course.

Signed on this, 30 October 2017, in Potchefstroom.

_____

Cordre Smith

Table of Contents

List of Figures

List of Table

List of Abbreviations

BJT	Bipolar Junction Transistor
CC	Constant Current
CS	Chip Select
CV	Constant Voltage
DMIPS	Dhrystone Million Instruction per Second
DUT	Device Under Testing
GPIO	General-purpose Input/Output
GUI	Graphical User Interface
IDE	Integrated Development Environment
LCD	Liquid Crystal Display
MSPS	Mega-Samples per Seconds
Op Amp	Operation Amplifier
PCB	Printed Circuit Board
PSS	Power Supply System
PWM	Pulse Width Modulation
RMS	Root Mean Square
RTOS	Real-Time Operating System
SMPS	Switch Mode Power Supply
SPI	Serial Peripheral Interface
UI	User Interface

List of Symbols

V	Voltage
A	Ampere
m	milli

1. Introduction

1.1. Background

All of us know the feeling of sitting in the electronics lab in the early hours of the morning trying to figure out why your circuits are dysfunctional while starting your 4th rebuild of the night! How would it feel to do all this in the comfort of your own home? If the university was able to give you a small, cheap and portable power supply, this could be achieved, and you have your power supply.

This is accomplished by purchasing cheap single channel fully analogue power supplies from China. These supplies will cost approximately R1000 and most probably cheaper if bought in large quantity. However, these supplies are rather heavy, weighing in the region of 5 kg [1]. The other option for the university is to buy programmable power supplies, these can also be bought online and in big batches, but they are rather expensive with an approximated cost of R7000 + [2].

Thus the optimal solution would be designing a cost-effective, a portable power supply that is user-friendly, provides basic functionality, thus displaying constant current and constant voltage capabilities, and providing the student with the opportunity to do off-campus development.

1.2. Problem Statement

The problem that is addressed is to develop a cost-effective portable power supply that can present the basic features to a student. It should serve the purpose of giving the student better accessibility and comfortability while working on projects. To comply to the user-friendliness of the project the power supply is controlled by a Graphical user interface. Thus implying that the power supply is digitally controlled.

To comply with the cheap aspect the power supply should be completed within a budget of R2000 or less, it should also be easily portable with an easy and fast setup and be light enough for an engineering student (female/male) to carry.

The digital aspect of the powers supply (that is implemented with the use of a microcontroller and some form of a graphical user interface) will remove the need for an analogue turn knob, thus increasing the precision of the power supply considerably. While also providing a user-friendly interface to work with.

1.3. Project Scope

The scope of this Final Year project is to design a cheap, portable digital power supply that is provided to future first-year students, it consists,

- of developing software that is implemented on some microcontroller,
- developing and constructing the electronic parts of the power supply,
- developing some form of graphical user interface,
- should be portable for any gender,

The scope of the project does not include,

- developing the microcontroller,
- building components from scratch,
- worrying about the AC output of the wall socket,
- controlling the environment of where the device is used,
- considering students with different fields of study, the user in this scope is an Engineering student,

The implementation phase should be completed by demonstration day, 25 October 2017, and the final report on 30 October 2017.

1.4. Project Objectives

The project has some main objectives that should be meant, for it to be seen as successful. These objectives are listed below,

- It should be able to be built cheaply,
- It should be portable,
- It should present a Graphical User Interface (GUI)
- It should be an AC/DC converter
- It should have Constant Current capabilities
- It should have Constant Voltage capabilities

1.5. Methodology

The methodology followed throughout the project is the Systems Engineering methodology. The flow diagram of the methodology is shown in Figure 1 [3].

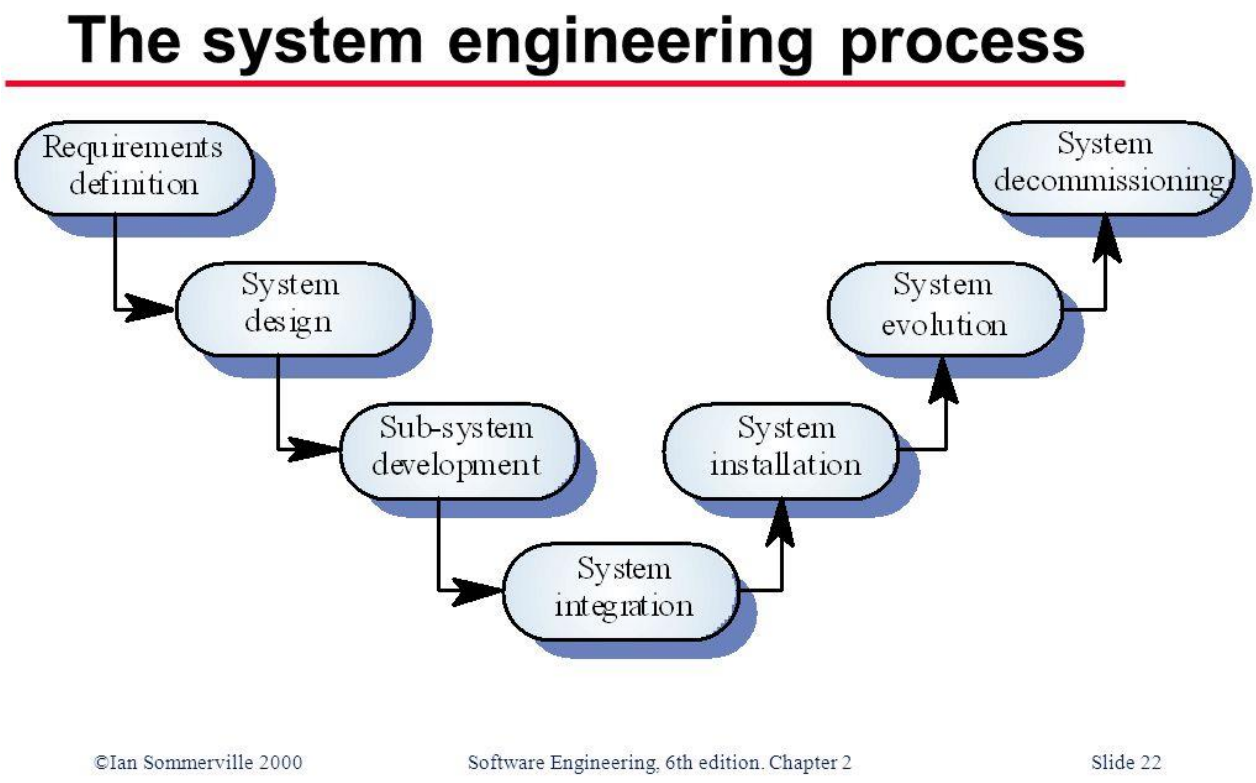


Figure 1: Systems Engineering Methodology

From the document that follows it is seen that this process was properly followed and each step was executed up and until System installation.

1.6. Document Overview

This document will consist of 6 main chapters, of which this is the first one, the following chapter is,

- Literature Study, Chapter 2
- Design, Chapter 3
- Implementation, Chapter 4
- Testing and Evaluation, Chapter 5
- Conclusion and Recommendation, Chapter 6

2. Literature Study

2.1. Introduction

In Chapter 1: Introduction the problem is stated, but for a quick refresher, this project aims to build a cheap, portable, programmable, digital power supply that is given to first year Engineering students to take home with them. That serves the purpose of giving the student better accessibility and comfortability while working on projects.

In Figure 2 the basic context diagram of the system is shown. The arrows of the context diagram represent different forms of interaction. The red is electricity, the orange is heat transfer, the green is visual, and the blue is touch. For this literature study, the different parts of the system are taken apart and evaluated separately.

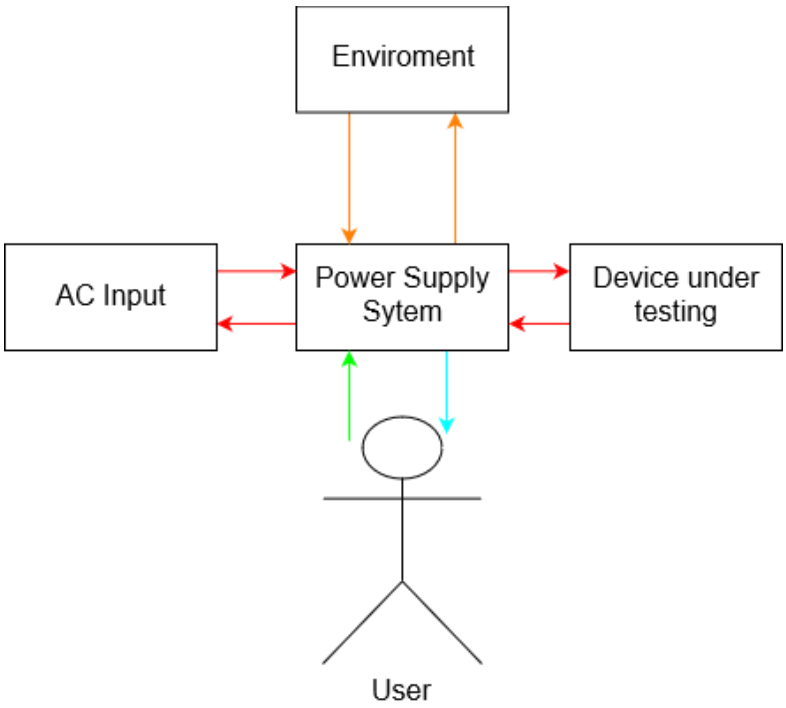


Figure 2: Context Diagram of the System

In the following sections, the different parts of the system shown in Figure 2 are looked at individually. In Section 2.2 the AC Input from ESKOM is looked at with Sections 2.3 to 2.6 covering the remaining 4 parts of the context diagram.

2.2. AC Input

The AC Input of the device is received from a three-prong household wall socket. The rating of the output of these wall outlets according to [4] is,

- 220 $V_{RMS;}$
- 16 $A_{Max;}$
- 50 Hz.

As said in Chapter 1, the AC Input that is received from the wall socket and indirectly ESKOM is out of context for this project and is considered a set constant with the scope of the project.

The Power Supply System (PSS) is designed not to supply energy back to the power grid. Thus the interaction with the PSS on the power grid is to be taken into consideration at the start.

2.3. Environment

The environment is a typical room (hostel/apartment/student home) in Potchefstroom, the average conditions according to [5] are,

- Room temperature is 16 ºC;
- 51 % relative humidity.

As said in Chapter 1, controlling the environment where the device is used is not in the context of the project, and the standardised values assigned above is accepted as the operating conditions.

There is heat generated will the device is running thus it will affect the surrounding environment. However, this effect is minuscule and is ignored.

2.4. Device under testing (DUT)

The device under testing is the electronic component that the student wants to test, from perusal of practical requirements it was found that the DUT of a typical university practical has,

- Operating voltage of 3.3V - 30 V;
- Operating a current range of 10mA – 3 A.

The student will also need the power supply to exhibit Constant Current (CC) and Constant Voltage (CV) capabilities; CC is for protection against overcurrent, one of the highest causes of the sudden explosion of student electronics.

The DUT only uses the amount of current that it needs and not all the current that the user capped the PSS at in the start, for the PSS to know how much current to deliver the DUT has to “tell” or effectively draw current from the PSS and thus has an effect on the PSS. The PSS should be capable of dealing with this behaviour even though it is common since current cannot be forced into a system as voltage can.

Although the DUT is not in the scope, interfacing with the DUT is in the scope. This means that when the designing phase is implemented consideration towards this should be taken, and protection for overcurrent and surges should be added in the electronics.

2.5. User

The user in the scope of the project is assumed to be an Engineering Student and has the knowledge as to how a normal power supply function. This allows the project to be made more complicated without lots of manuals and documentation being have to draft up to teach the user how to use the PSS.

The user can also be male or female, thus impacting the portability of the PSS. The PSS should be able to be carried by a male or female student, and thought should be given to this when the portability or more precisely the weight of the PSS is designed.

The field of study of the student is not part of the scope of the project and is assumed to be an engineering student when the designing is done. However, the gender of the student is in the framework of the project and consideration towards this should be made when the design process is being implemented.

2.6. Power Supply System (PSS)

In Figure 3 a more zoomed in and detailed version of the Power Supply System is seen, as appears from this figure, the PSS consists of two main parts,

- Power supply Electronics;
- User Interface (UI).

To fully cover all the technologies available for implementation, the PSS is broken up into these two parts, and both are discussed in detail.

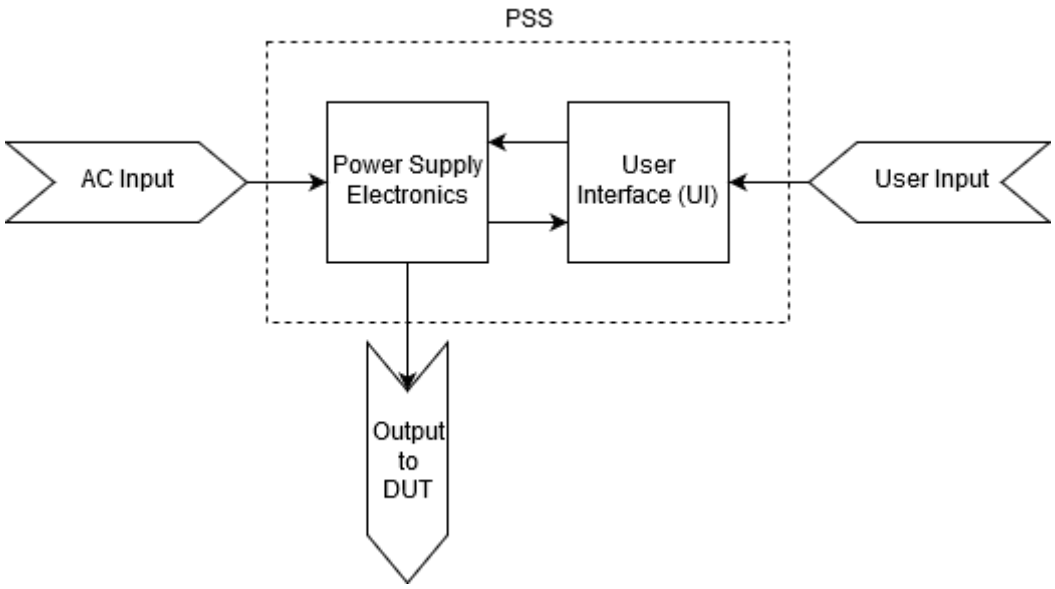


Figure 3: PSS Zoomed in Context Diagram

2.6.1. Power Supply Electronics

First of all, let's take a look at the two common types of power supply topologies that are used, they are,

- Linear Power Supplies;
- Switching-mode Power Supplies.

2.6.1.1. Linear Power Supplies

Linear Power Supplies commonly have a very basic concept with a straightforward design as is seen in Figure 4.

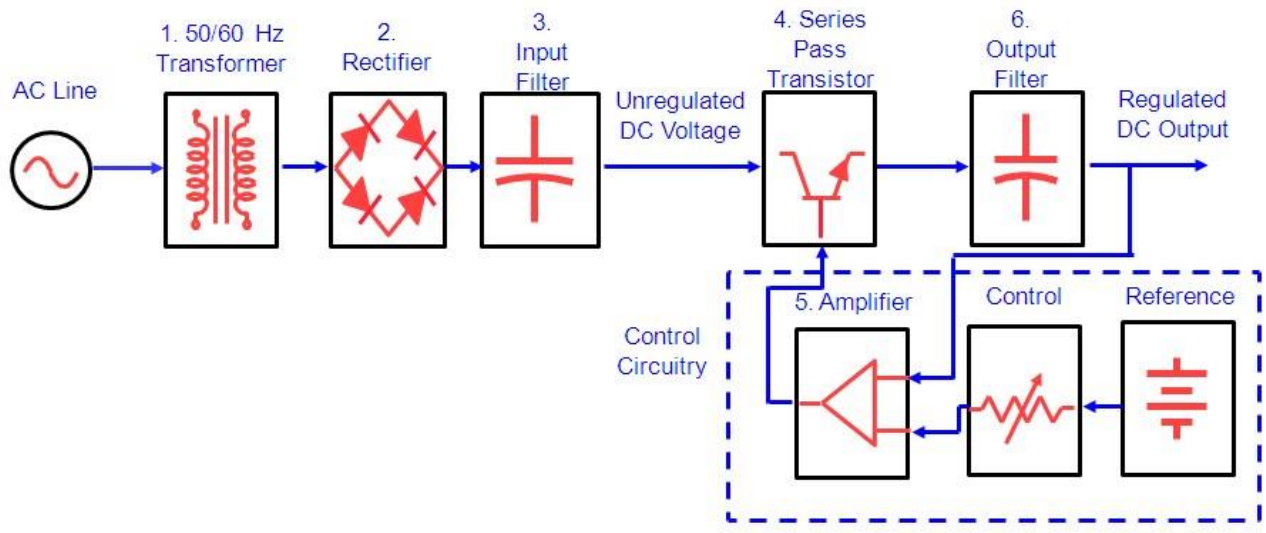


Figure 4: Linear Power Supply

Because of their simple design they also have basic, straightforward mechanics, this means that they have some very nice advantages [6],

- Fast transient response;
- Low output noise and ripple voltage;
- Cost effective at low power outputs.

However, due to the reasons above they have some inherent disadvantages,

- Relatively large and heavy, at high power outputs;
- Low power efficiency;
- Expensive when going to higher power outputs.

2.6.1.2. Switch-Mode Power Supplies

Switch-mode Power Supplies has more complexity compared to Linear Power Supplies but still works straightforwardly, in Figure 5 a basic implementation diagram of a switch-mode power supply is seen [7],

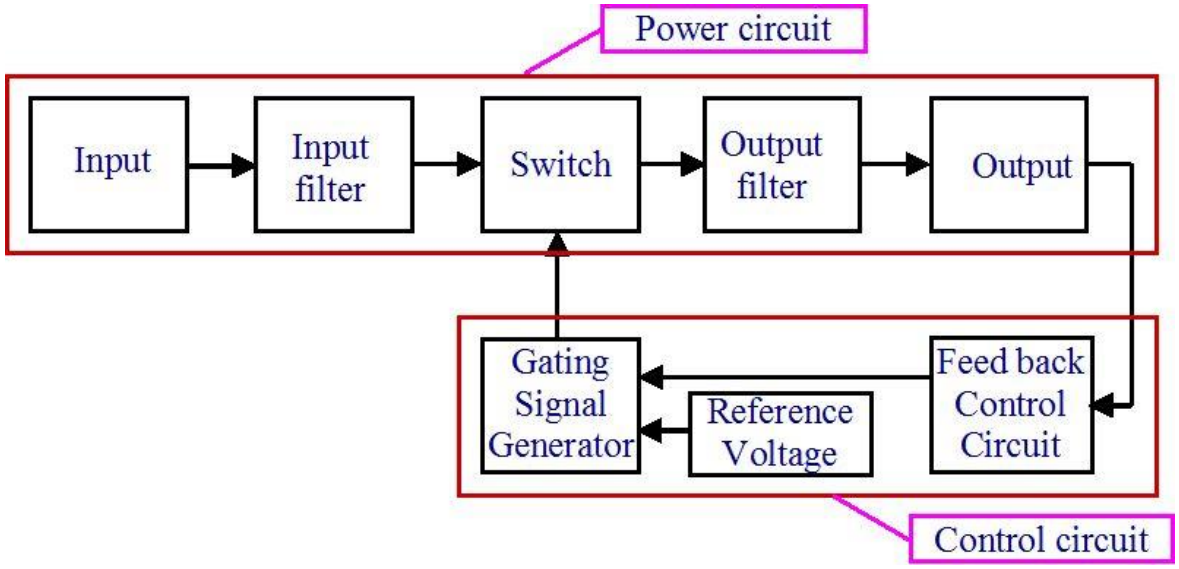


Figure 5: Switch-Mode Power Supply

Switch-mode Power Supplies have some nice advantages [8], shown below,

- High power efficiency;
- Small and lightweight, at high power output;
- Cost effective, at high power output.

They also have some disadvantages [8], shown below,

- High output noise;
- High ripple voltage;

- Slow transient response.

2.6.1.3. Conclusion regarding Power Supply Topology

The PSS that we are designing needs to comply with the following guidelines,

- The maximum output voltage should be 30 V
- The range of the output current should be 10 mA to 3 A
- The PSS is going to be used for lab implementation, meaning
    - Low output noise is needed
    - Low voltage ripple is needed

Because the output range of the PSS is going to be at a maximum of 30 V and 3 A, the maximum power the PSS is able to deliver is 90 W, thus meaning that the PSS falls into the category of low power output. The PSS also needs low ripple and output noise because it is implemented as a lab power supply that is power low power electronics. Taking these reasons into consideration, the Switch-Mode power supply is eliminated for our purpose; this leaves us with the Linear Power Supply Topology.

In the next session, he are taking an in-depth look at the Linear Power Supply topology that we are implementing.

2.6.2. Detailed Power Supply Design

In Figure 6 a detailed representation of the power supply topology that is carried out in this design is seen,

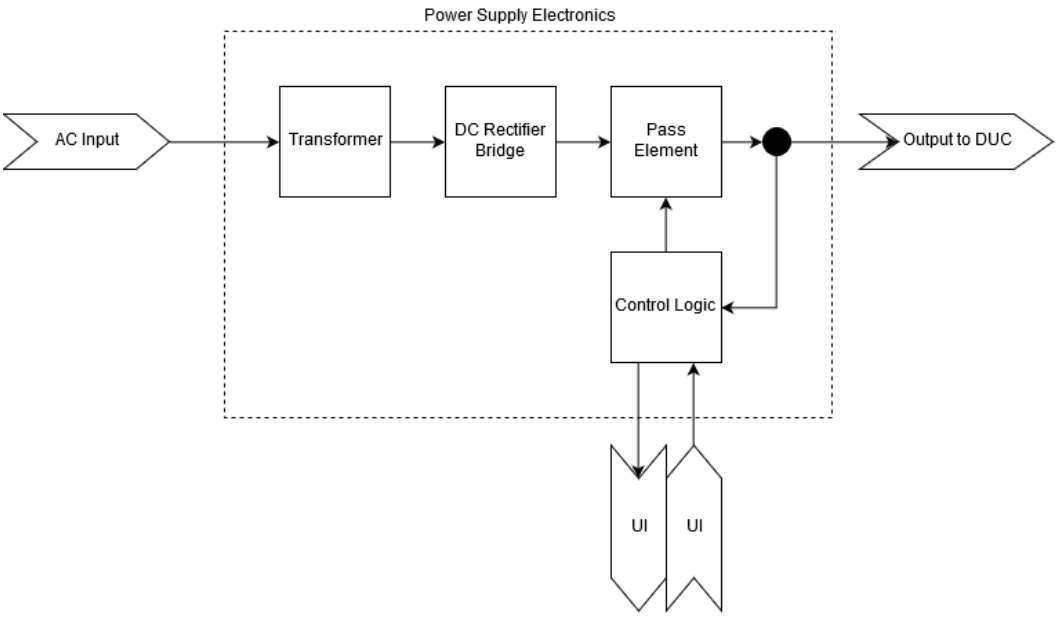


Figure 6: Detailed Context Diagram of Power Supply

The power supply is broken up into 4 main parts,

- Transformer;
- DC Rectifier bridge;
- Pass Element;
- Control Logic.

There are also 2 filter stages present, an input filter after the DC Rectifier Bridge and an output filter after the Pass Element before for DUC. The four stages are discussed in further detail in the following four subsections.

2.6.2.1. Transformer

In Figure 7 a diagram of a transformer is shown [9].

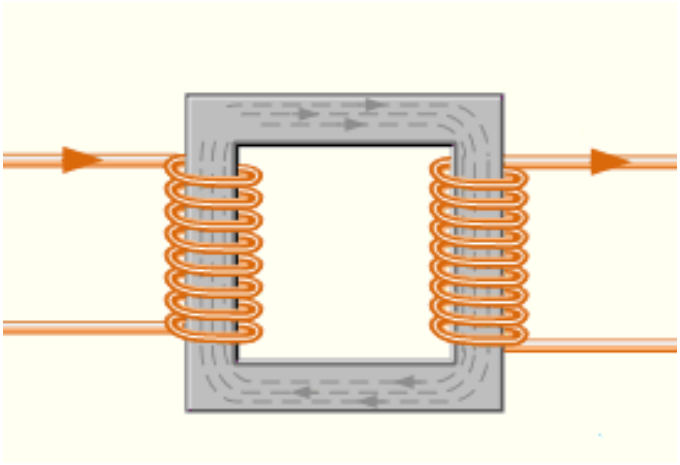


Figure 7: Example of a Transformer

The transformer that needs to be implemented needs to be a 50 Hz step down transformer. This is chosen as the required transformer because in the AC Input Section the AC input from the wall socket was standardised as 220 V_{RMS} and 50 Hz.

2.6.2.2. DC Rectifier Bridge

In Figure 8, the rectifier that is implemented is shown. This is a commonly used DC rectifier, and not much discussion around this is needed because other alternatives are not used in this scenario [10].

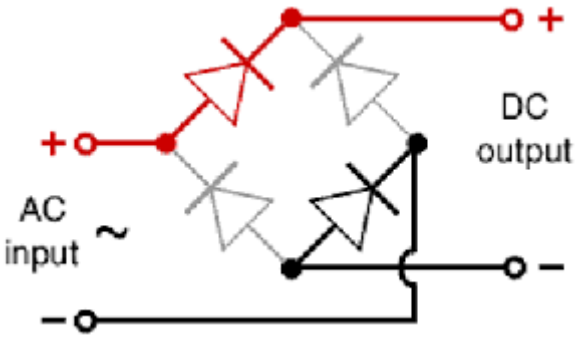


Figure 8: DC Rectifier Bridge

2.6.2.3. Pass Element

Pass Elements are variable resistance devices, power transistors, in series with the DC output from the DC rectifier. The error signal from the output is amplified by an operation amplifier and fed into the power transistor. The resistance of the transistor is increased when the output should go lower or decreased when the output should increase [11].

There are two types of Pass Elements that we can choose from,

- An integrated Pass Element;
- Discrete Pass Element.

2.6.2.3.1. Integrated Pass Element

Integrated Pass Elements are fully packaged Darlington pair with an Operational amplifier sold as one unit, the advantage of these are,

- The instability of these packages are predesigned and worked on by experienced analogue designers, thus meaning no instabilities will occur

The disadvantages of these Integrated Pass Elements are,

- They have set amount of Power they can dissipate

Thus, the package is easily bought, and the instabilities of the pass element are corrected by experienced designers, thus allowing the designer to focus on other aspects of the project, the only problem will come in when high amounts of power need to be dissipated.

2.6.2.3.2. Discrete Pass Element

What is meant by Discrete Pass Element is that the designer designs its pass element, in other words, they add their transistor and operational amplifier and develop a pass element, the advantage of using this is,

- Can dissipate as much power as the design needs, by adding more transistors to the design

The disadvantage of this design is,

- Instabilities are very hard to control in transistors and takes much work to try and configure and design correctly

Thus, if a high amount of power needs to be dissipated this is the way to go, however, the designer should be ready to figure out the instabilities of the transistors.

2.6.2.3.3. Conclusion regarding Pass Element

Taking into consideration that the PSS should only be able to dissipate 90 W at maximum and the main focus of this project is to develop a programmable aspect for the PSS, the Discrete Pass Element is disqualified as an option.

This gives the designer more time to focus on the actual main parts of the project by taking away the work of trying to stabilise the transistors.

2.6.2.4. Control Logic

The control logic is the part of the Power Supply that tells the Pass element how much the output should be, the control logic receives the input from the user and changes the reference voltage of the operational amplifier. This then changes the resistance of the power transistor of the pass element and in doing so changes the output value [12].

The control loop of the PSS is seen in Figure 9.

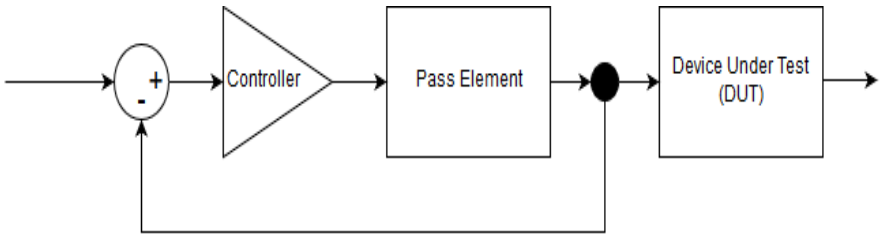


Figure 9: Power Supply Control Loop

There are four types of controllers that are implemented, these are,

- Analogue;
- Digital Signal Processor (DSP);
- Computer;
- Microcontroller;

- Discrete Electronics.

In the following four subsections, these 4 types are discussed and compared.

2.6.2.4.1.Analogue

If analogue is implemented the reference voltage of the operational amplifier of the pass element is connected to the output from a variable resistor. The resistor value is changed by turning a knob, and this then changes the reference voltage that in turn changes the output voltage of the PSS. This method is tried, tested and unbelievably well documented; it is easy to implement due to all the documentation and carries the advantages of being,

- Extremely fast
- Low complexity, thus fewer places for errors to appear

The analogue has one key disadvantage; it cannot be used to implement programmability features. This is one of the key features of the project, and thus an analogue controller does not suffice in our design.

2.6.2.4.2.Digital Signal Processor (DSP)

Digital Signal Processors is a processor that modifies analogue of digital signals according to certain mathematical and computational algorithms. These devices are most commonly used to improve signals and output a signal of higher quality. However, these can also be used, with the correct algorithms, to achieve the operations that we are after [13].

Thus, by implementing a DSP the digital and programmability aspect of the PSS is accomplished. However, there are some problems when trying to achieve this. DSP's are extremely fast when working on one channel of duplex communication, but when more than one channel needs to be communicated at a time, it has trouble keeping up with the demand. In the PSS the DSP has to receive the following inputs,

- Error signal from output
- The output wanted from the user

Moreover, has to provide the following outputs,

- The current output of the PSS
- The reference voltage of the operational amplifier of the pass element

These interactions are seen in Figure 6.

Thus taking into consideration that there are multiple channels that should be interacted with, a DSP is disqualified as a viable solution for the PSS.

2.6.2.4.3.Computer

One of the other options there is when trying to decide what type of Controller to implement, is implementing a desktop or laptop computer as the controller of the system. This reduces the cost of the system because no money has to be invested into buying a controller. This then makes it easier to comply with the cheap aspects of the project requirements.

There are two downsides to implementing a computer as the controller, these are,

- The PSS will not be able to be developed as a complete module because a computer or laptop will always have to be present for the PSS to function.
- Extra software has to be installed on the computer or laptop
- The reference voltage of the operational amplifier has to be updated on a very short interval; this is difficult for the computer to achieve because it has to receive the data, interpret it and then resend the new value within a tiny amount of time.

Thus, to use a computer as a controller is plausible but it is not the best method of achieving it, other methods are better to implement.

2.6.2.4.4. Microcontroller

The final option that is available is implementing a microcontroller this will add complexity to the design because Analogue to Digital (ADC) and Digital to Analogue converters are going to have to be implemented to allow the microcontroller to communicate with the rest of the electronics.

Implementing a DAC will not be a problem, a common variety DAC has a sampling rate of 500 MSPS [14], where MSPS stands for Mega-Samples per Second this means it can easily measure a signal of 250 MHz, this is half the sampling rate of the DAC because of Nyquist-Shannon sampling theory. This means that a common variety DAC is able to work for our application.

Implementing an ADC will also not be a problem, a common variety ADC has a sampling rate of 20 MSPS [15], and this means the ADC can easily interpret a 10 MHz signal. Thus meaning it will easily suffice for the PSS.

Thus after getting the ADC and DAC out of the way, let’s have a look at the microcontrollers that is implemented, an example of a microcontroller is seen below in Figure 10 this is a microcontroller for STMicroelectronics,

Figure 10: ST Microcontroller

When looking at implementing a microcontroller it is easier to implement a developer board, this removes the need to do soldering of the pins of the microcontroller and adds extra functions that can apply to the PSS. In Figure 11 an example of a commonly used developer's board the STM32F407VG is seen [16],

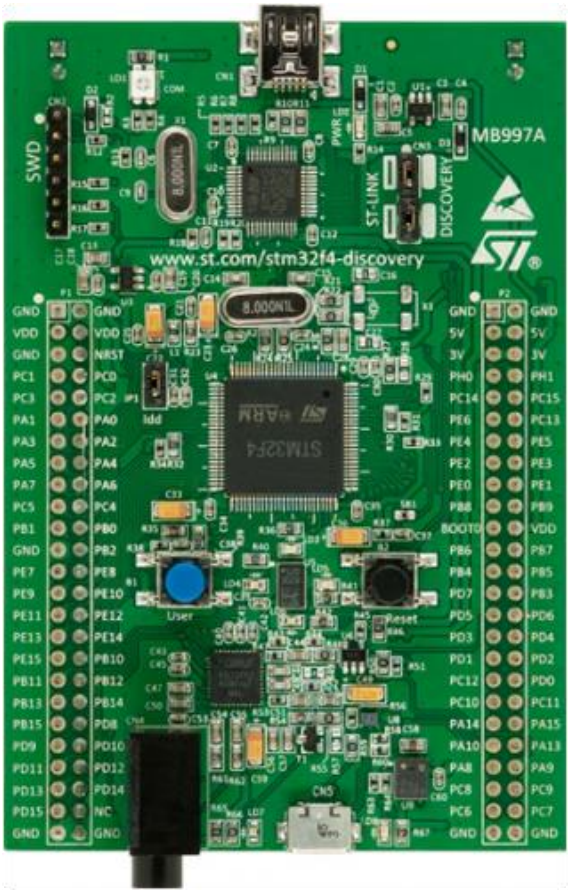


Figure 11: STM32f407VG Developer Board

The developer's board shown above has some applicable features that are useful when implementing our PSS, the specifications of the developer's boards shown above are [17],

- ARM 32-bit Cortex M4 Microcontroller;
- Frequency of up to 138 MHz;
- DSP Instructions;
- LCD parallel interface;
- 2x12 Bit D/A Converters;
- 140 I/O Ports.

These features listed above shows that a developer board might be a good option to consider when making a design choice as to what controller to implement.

The microcontroller mentioned above and developer board is just used as an example to give a perception of how valuable one of these might be in the development of the PSS. In the following section, a detailed discussion of the available microcontrollers is done.

2.6.2.4.4.1. Types of Microcontrollers

The Microcontroller that is being considered has been narrowed down to ST's products; this is due to familiarity with the products and availability. To have a broad perspective of the available microprocessors from ST, a detailed look at the M3 M4 and M7 ARM, 32-bit Cortex processors, based MCUs is done in Table 1 [18],

Table 1: MCU Comparison

Processor	MCU	Speed (MHz)	DMIPS/MHz
M3	STM 32 F1	72	61
	STM 32 F2	120	150
M4	STM 32 F3	72	90
	STM 32 F4	180	225
M7	SMT 32 F7	216	462

2.6.2.4.5. Electronics

The other option for a controller that is used is by implementing an electronic controlling circuit that is driven by a PWM output or a DAC output from a microcontroller. This option reduces the load on the microcontroller and thus open up options regarding User interfaces.

The most common controlling circuits are voltage regulators. In Figure 12 a basic voltage control circuit is seen. The circuits are shown in Figure 12, and Figure 13 are discrete implementations of these types of circuits, this is not the only option that is available. These parts can also be replaced by integrated solutions or chips.

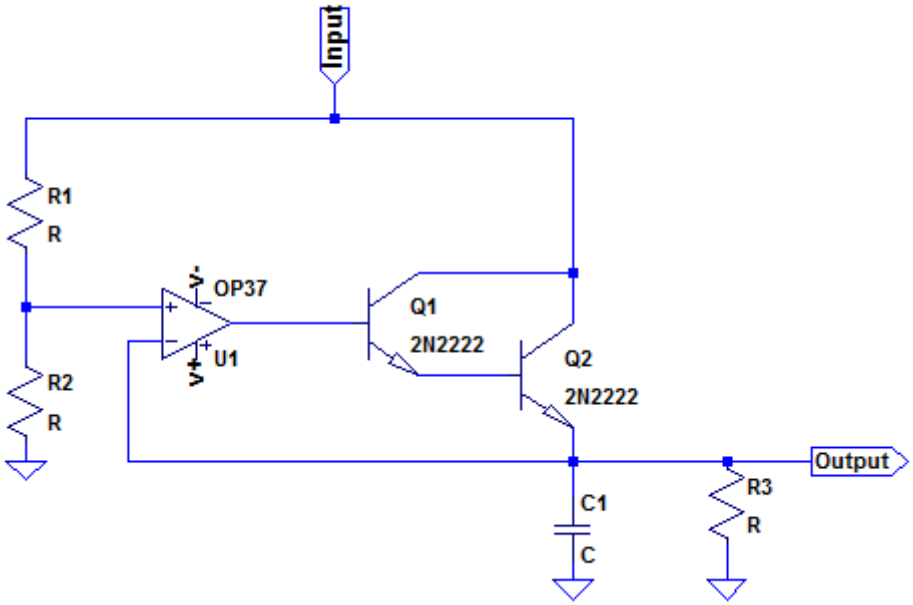


Figure 12: Voltage control circuitry

In Figure 13 a basic current control circuit is seen.

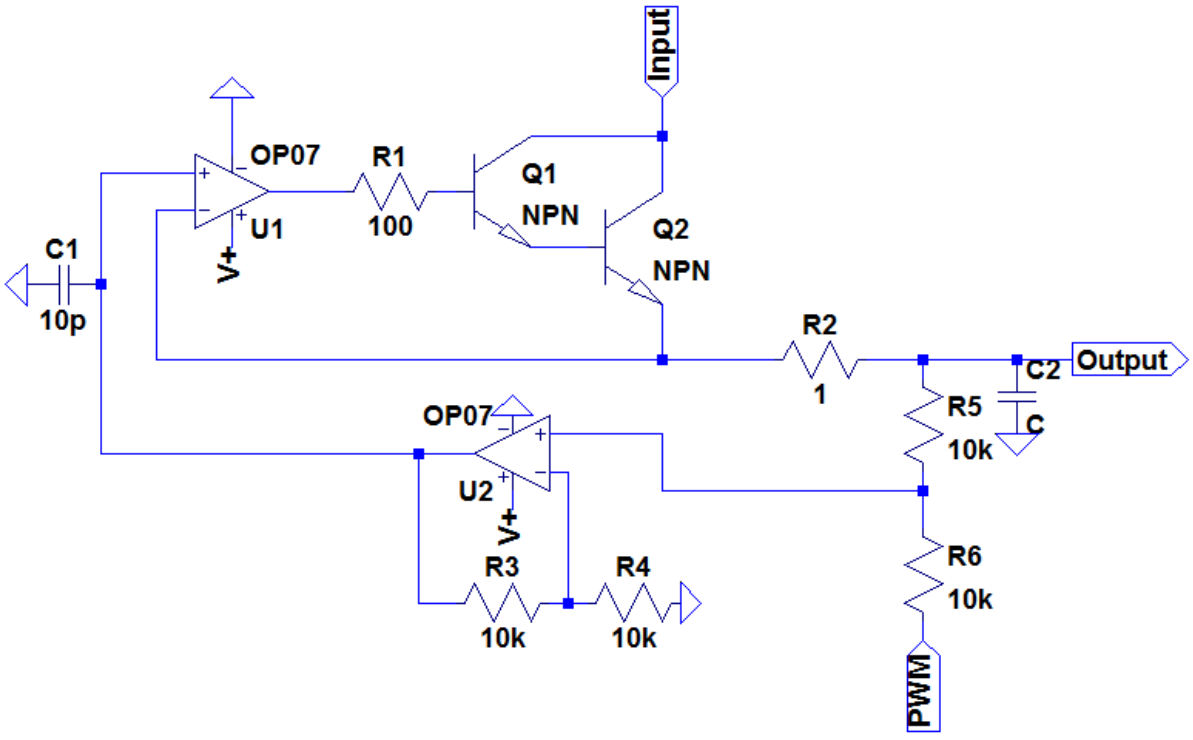


Figure 13: Current control circuit

2.6.2.4.6. Conclusion regarding the Controller

As said in the various subsections, the Analogue, DSP and Computer controllers are disqualified for various reasons, and thus the choice came down to a Microcontroller. In the microcontroller section, it was shown that DAC and ADC would be able to keep up with the required frequency. The was

also comparison done between various MCUs, and all of the data available is used to make an informed and well thought through decision about what the MCU is that should be implemented as the controller.

2.6.3. User Interface (UI)

There are different methods of implementing UI's for power supplies, in the list below a few of them are mentioned,

- Buttons with no Display;
- 7 Segment Displays with Knobs;
- LCD;
- Computer interface.

The 7 Segment with Knobs is the most common and is found in all the cheap variety power supplies. However, the PSS should be digital and programmable thus with these constraints in mind the first two options are automatically disqualified. Because we are looking at implementing a Graphical User Interface or more commonly known as a GUI. Even though a 7 Segment has some graphical capabilities, these are far too low for the output we want to achieve. Thus we are left with two options,

- LCD;
- Computer Interface.

A closer look at these two options is done in the following sections.

2.6.3.1. LCD

Liquid Crystal Display or LCD are the screens that are commonly found on your phone and tablets. These display like LED displays are very thin and doesn't take up to much space or weight [19]. In Figure 14 an example of a commonly found LCD is seen. They come in a massive variety of sizes and is bought quite cheaply with some searching.

Figure 14: Common LCD

The advantages of implementing an LCD in the project are,

- Thin and light;
- Touchscreen, thus allowing nice displays to be created;
- Is written to at a very fast tempo;
- The PSS is completed as a complete module.

The disadvantages of LCD's are rather limited and mostly based on where they are implemented if implemented in our project these disadvantages are,

- If an LCD is implemented a microcontroller needs to be used for control, thus meaning the controller that is going to be implemented should be faster than previously thought or an increased number of controllers should be implemented
- The software used to interface with an LCD takes some time to develop, thus increasing the time it takes for the implementation phase to be completed

2.6.3.2. Computer Interface

What is meant by a Computer Interface is that the PSS plugs into your laptop or desktop via a USB cable? The data from the PSS (Current Outputs, Graphs, etc.) gets displayed on your computer screen. The output voltage and current can then also be adjusted from your desktop. The advantages this has are,

- Zero cost to implement;
- The display is made quite complex;
- No need for extra or faster controllers to be able to run the graphics updates.

This method also has its disadvantages, and these are,

- Laptop or desktop computer needs to be present when trying to use the PSS;
- Extra software needs to be installed on your computer;
- Risks of being slow to respond to user interaction;
- Can’t make the PSS a complete module.

2.6.3.3. Conclusion regarding User Interface

There is some vast difference between the two methods. However, there is some difference that is more important in our regard. The fact that the Computer Interface costs no extra money to implement improves the chances of creating a cheap PSS. The other glaring difference is the fact that when a computer interface is implemented the PSS cannot be developed as a complete module because a computer is needed to operate the PSS. These two are the major difference between the various methods it should be thoroughly thought through when deciding on the final design.

In the above section, enough information has been given to be able to make an informed decision, on a later stadium, regarding the GUI that is going to be implemented in this design.

2.7.Software

The main software issue that needs to be addressed is the type of operating system that is implemented. Two main operating systems are being considered these are mbed and FreeRTOS in Table 2 the comparison of these two operating system is seen.

Table 2: Comparison of OSs

	Difficulty	Performance	Implementation	Integration	Total
FreeRTOS	5	8	8	7	28
mbed	7	8	9	10	34

In this section, there is enough information regarding the operating systems to be able to choose the operating system that will work best with the intendant solution at a later stadium.

3. Design

3.1. Introduction

In Chapter 1: Introduction (pg. 1) the purpose and reason for the project were laid out, together with the project scope and objectives. In Chapter 2: Literature Study (pg. 4) a detailed explanation of the possible solutions and implementable equipment is given. In this chapter (Chapter 3: Detailed Design) the design choices and final design (architecture) of the intendant solution are shown with an in-depth explanation as to why the choice where made.

As stated in the previous paragraph the scope, objectives and requirements of the project are listed, thus considering these factors a detailed plan as to what exactly the project should be able to accomplish should be the first step of the design process.

In Section 3.2.1 the states and modes diagram is shown, this is the first part of the design process and shows, as implied by the name, the various states that the project should have and the modes that are allowed within these states, with the interactions between the states and modes laid out.

In Sections 3.2.2 and 3.2.3, the Operational and Functional analysis that was done is shown. This analysis was done to narrow down the exact interactions the user has with the project (Operational analysis), and the working process of the project's electronic and programming aspects (Functional analysis) to achieve the objectives of the project and allows for the user interactions and operations shown in the Operational analysis to occur. This is done to narrow down the exact processes the project should be able to complete and this in return gives us a more detailed look at to exactly what the project should be able to accomplish to satisfy the project requirements.

From these 3 sections, the exact project accomplishments are determined, and it can now be used to design the first iteration of the architecture, this architecture is shown in Section 3.2.4. This architecture shows the basic parts of the project at a very high level. Section 3.2.5 (Resource allocation) is used to show that all the functions shown in the functional analysis are done by the components present in the Architecture and that no extra parts are included.

The preliminary part of the design process was done on a very high level and was used to firstly narrow down exactly what the project should be able to do and secondly, to show what parts and processes are needed to achieve the requirements of the project.

The detailed design stage is a lower level look at the functional analysis and architecture. The functional analysis that was done and shown in Section 3.2.3 is taken, and each block that is shown in that analysis is broken down further until the blocks cannot be broken down anymore. This shows a highly in-depth look at the exact operation of the system with all the functions needed and how

exactly they are implemented. The implementation of this together with the results is seen in Section 3.3.1.

By completing the expanded functional flow, it comes obviously as to what coding algorithms and coding needs are needed. These algorithms together with some trade-offs regarding the coding aspects of the project are shown in Section 3.3.2.

In Section 3.3.4, the detailed architecture is shown. This architecture was done with the intent to show a detailed look at all the components that were chosen through the trade-offs, all the trade-offs that have been made not regarding software is shown in Section 3.3.3. It is also used to show what type of communication channels is needed and what voltage levels the different systems are working on.

Finally, in Section 3.3.5 a detailed resource allocation is done that show that the solution is viable and that all the requirements of the project have been met.

This Chapter shows that a thoughtful and comprehensive Engineering design process has been followed and executed correctly.

3.2. Preliminary design

3.2.1. States and Modes

In Figure 15 the states and modes diagram is shown. It is seen from this that there are two states. The operational state and non-operational state, for this project a manufacturing state will not be applicable due to only one PSS being built.

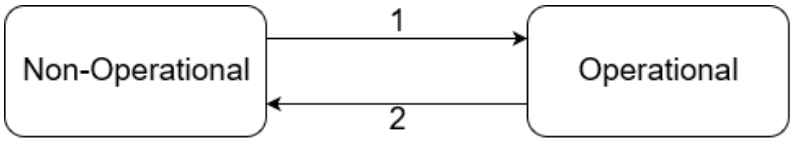


Figure 15: States and Modes diagram

From Figure 15 two interactions (no 1 and 2) are visible these are from operational to non-operational and vice versa. The interactions are applicable in the following situations,

1) When the output of the PSS is turned off, the PSS is switched from operational to non-operational.
2) The PSS is switched from non-operational to operational if the PSS is plugged into a wall socket.

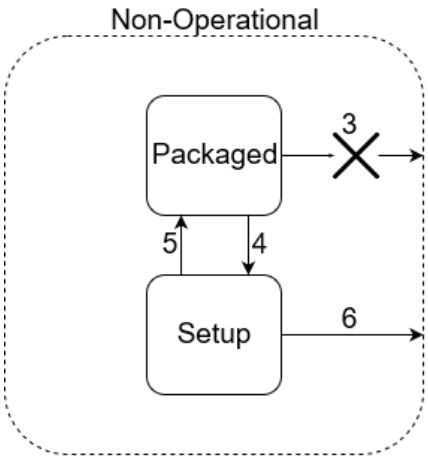


Figure 16: Non-operational state's modes

In Figure 16 the non-operational state's modes are seen. The packaged mode applies to this project because the PSS should be transportable. The modes also have transitions between them, and they are only applicable in certain situations,

3) It will never be possible to switch from non-operational to operational when the PSS is in the packaged mode.
4) From packaging to setup is achieved by plugging the PSS into the wall.
5) From setup to packaged is achieved by unplugging the PSS.
6) When setup is done properly, the PSS is switched to operational.

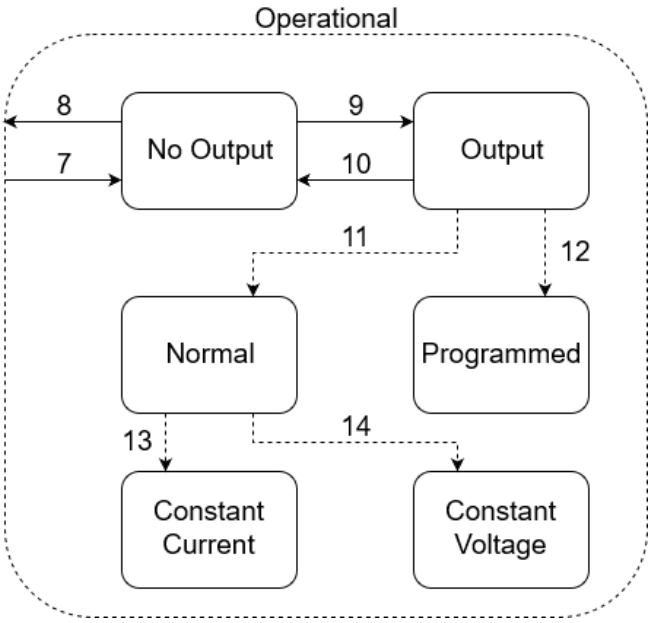


Figure 17: Operational state's modes

In Figure 17 the operational state's modes are seen. Similar to the previous modes of the non-operational state there are transitions between the modes and they are only executed when,

7) When the PSS is switched on, via the main power button, the PSS is set into no output mode.

8) The PSS is only allowed to switch from Operational to Non-Operational when the PSS is in no output mode.

9) The PSS is switched from no output mode to output mode via the output button; this is only applicable when the outputs of the PSS has been configured, and a DUT has been connected.

10) The PSS is switched from output mode to no output mode via the output button.

11) When the PSS is switched to output mode, it switches to normal mode when no programmed output was configured.

12) When the PSS is switched to output mode if a programmed output is configured it switches to program mode.

13) The PSS switches to the constant current mode from normal output mode if the maximum current that was set is reached.

14) The PSS switches to constant voltage mode from normal output mode id the maximum voltage that was set is reached.

The above-mentioned transitions together with the states and modes give a general impression as to what the PSS should be able to do and when it should be able to do this, it also shows some of the safety mechanisms that should be implemented.

3.2.2. Operational analysis

In the previous section, the operations that the PSS should be able to accomplish has been laid out. In this section the flow of how the user will interact with the PSS is laid out in an operational flow diagram, this diagram is shown in Figure 18.

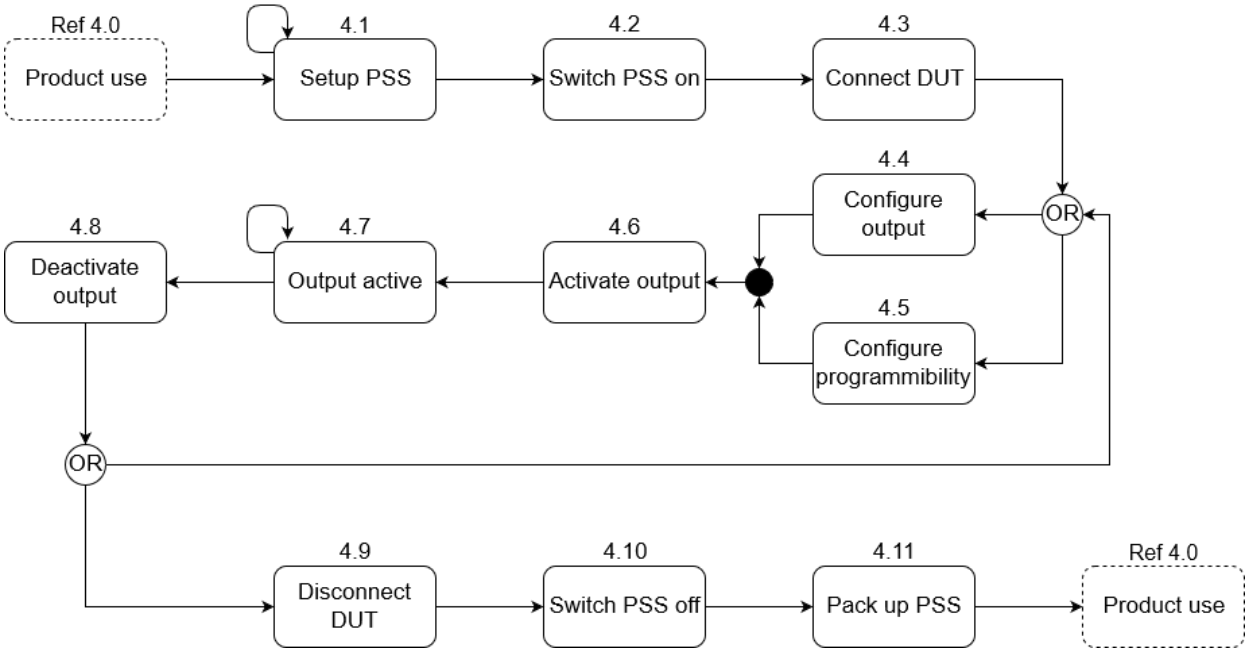


Figure 18: Operational flow diagram

From section 3.2.1 some interfaces are implied, these interfaces are,

- Power connection from wall socket;
- Main power button;
- Main output button;
- DUT connections to PSS;
- User interface (for programming outputs);
- The user interface (visual indications of output levels).

In Figure 18 all of these interfaces are used by the user to interact with the PSS. This shows that all the interfaces described in the previous section is not just applicable but they are all used, and no unneeded interfaces are described in the states and modes diagrams or no needed interfaces have been left out.

3.2.3. Functional analysis

From the previous section the operational analysis of the system has been completed, and in Figure 18 the operational flow diagram of the system is shown. The parts of the operational flow that have underlying device operations are discussed with a functional flow diagram provided.

The first part where the functional flow of the PSS comes into consideration is the start-up part of the PSS; this is block 4.2 in Figure 18. During the start-up of the PSS, the different threads of the program are started, the current state of the program is set to *no-output,* and the GUI is displayed (a mock-up display of the GUI is seen in Section 3.3.2). In Figure 19 the start-up routeing of the PSS is shown.

Figure 19: Start-up routine functional flow

The next functional flow, displayed in Figure 20, is the output configuration when normal output is desired. This is where the user sets the maximum output voltage and current, and then the PSS delivers constant current or constant voltage depending on the current output levels.

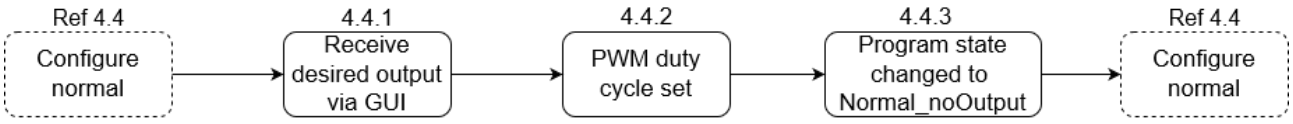


Figure 20: Output configuration functional flow

As is seen from Figure 20 the output desired by the user is set using the GUI, these values are then used to set the output levels of the DAC's that is controlling the set points of the PSS (this process is discussed in further detail in Section 3.2.4 and 3.3.2). The programs state is then changed to *normal no-output*.

Displayed in Figure 21 is the functional flow for the programmability output mode of the PSS. Two options are available to the user, firstly to choose from preconfigured outputs patterns or to program his custom output pattern. This pattern will then be saved, and the output level of the DAC's will then be set to the starting set points (as stated earlier the process is discussed in deeper detail in Section 3.2.4 and 3.3.2) and the state of the program is set to *programmed-no-output*.

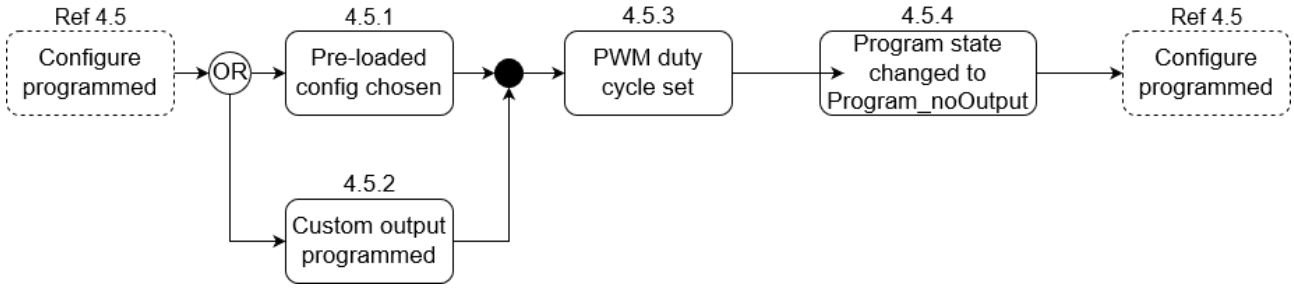


Figure 21: Programmability configuration functional flow

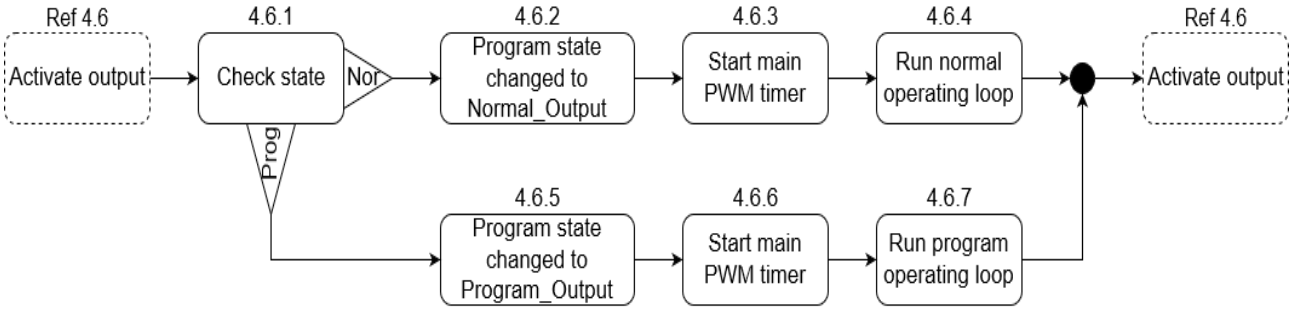


Figure 22: Activate output functional flow

The activate output's (block 4.6 in Figure 18) functional flow diagram is shown in Figure 22. Firstly the state of the PSS is checked and then acted upon. If the state is detected as *normal no-output,* the state is set to *normal output* and the main timer of the PWMs are started. The program then goes into a loop until the output is deactivated (block 4.7 Figure 18).

When the state is detected as *programmed no-output,* the state is set as *programmed output*. The main timer is then started, and the program goes into the programmed output loop until the output is deactivated (block 4.7 Figure 18).

In Figure 23 the deactivating of the output's functional flow is shown. When the main output button is pressed (and the system is in output state) the loop mentioned in the previous paragraph is broken, and the main timer of the PWMs are stopped. The duty cycle of these PWMs are also changed to zero, and the state of the program is changed to *no-output*.

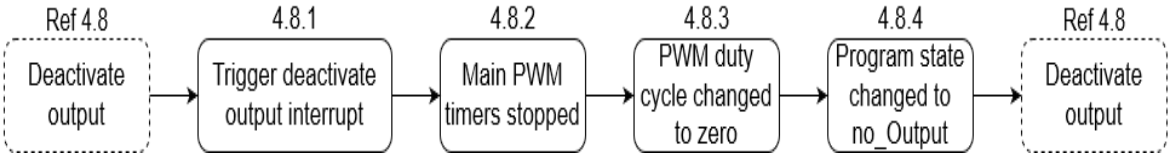


Figure 23: Deactivate output functional flow

This concludes the functional flows of the operational diagram shown in the previous section. The different states that the system is in has now been narrowed down together with all the functions of the system. The functions together with the interfaces are now used to develop an architecture for the system.

3.2.4. Architecture

The details of the project that was discussed and narrowed down in the previous sections are used to develop the architecture that is shown in Figure 24.

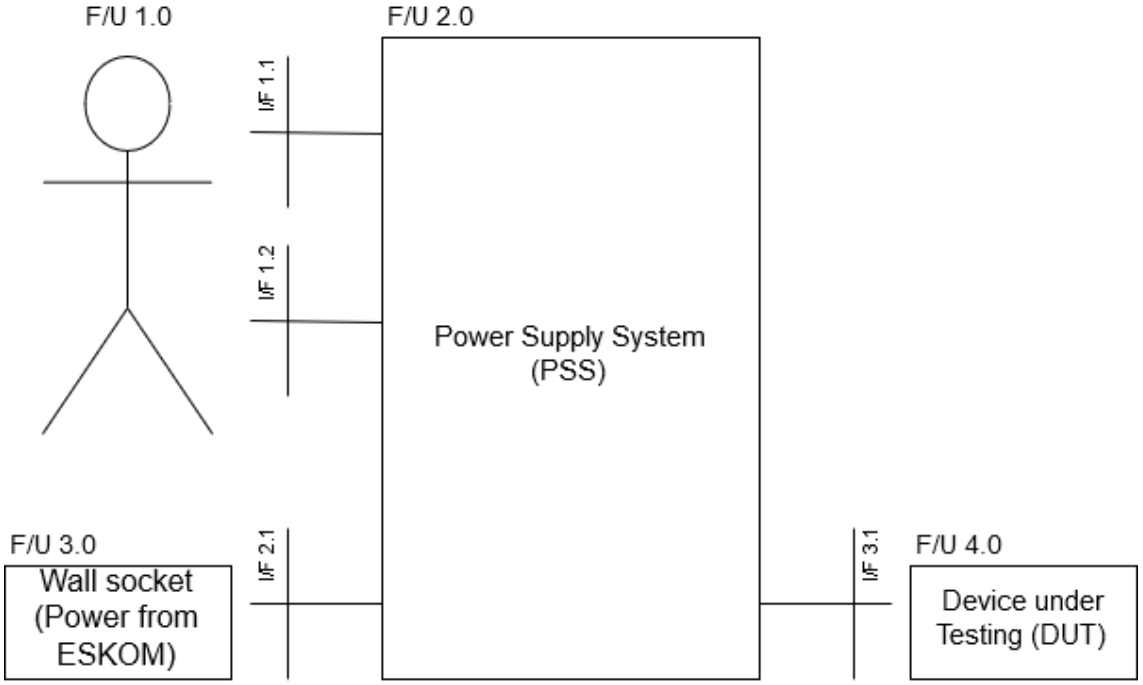


Figure 24: Basic architecture

3.2.5. Resource allocation

The functions listed in the operational analysis that is shown in Section 3.2.2 and shown in the operational flow shown in Figure 18 together with the functional units shown in the architecture in Figure 24 is used to develop the first basic resource allocation to determine if all the functions that are required can be accomplished by the current architecture. The resource allocation is seen in Table 3.

Table 3: Basic resource allocation

	Unit 1 : User	Unit 2: PSS	Unit 3 : Power Outlet	Unit 4: DUT
4.1 Setup PSS	✖	✖	✖	
4.2 Switch PSS on	✖	✖		
4.3 Connect DUT	✖	✖		✖
4.4 Configure Output	✖	✖		
4.5 Configure Program	✖	✖		
4.6 Activate Output	✖	✖		
4.7 Output Active		✖		✖
4.8 Deactive Output	✖	✖		
4.9 Disconnect DUT	✖	✖		✖
4.10 Switch PSS off	✖	✖		
4.11 Pack up PSS	✖	✖	✖	

Taking a look at the resource allocation, it is seen that all functions are achieved by one of the functional units described and that all functional units described have been used for one or more functions. This shows that the design is viable to this point and that no design errors have been made.

When taking another look at the resource allocation, it is seen that functional unit 3 or the PSS is used in all of the functions, this shows us that the current design should be expanded and that the trade-offs should be completed. This means that the preliminary design stage has been completed and that the following part of the design has to be the detailed design stage.

3.3. Detailed design

As stated at the end of the previous section the next part of the design algorithm is the detailed design section, this entails doing expanding on previous functional analysis together with expanding the current architecture. To do be able to expand the architecture some trade-offs have to be done, and a final resource allocation is needed to check the feasibility of the project.

3.3.1. Expanded functional analysis

The first functional expansion that is done for the functional flow shown in Figure 22 is done for *Run normal output operating loop* (block 4.6.4). This block is expanded, and the loop is developed. The results of this expansion are seen in Figure 25.

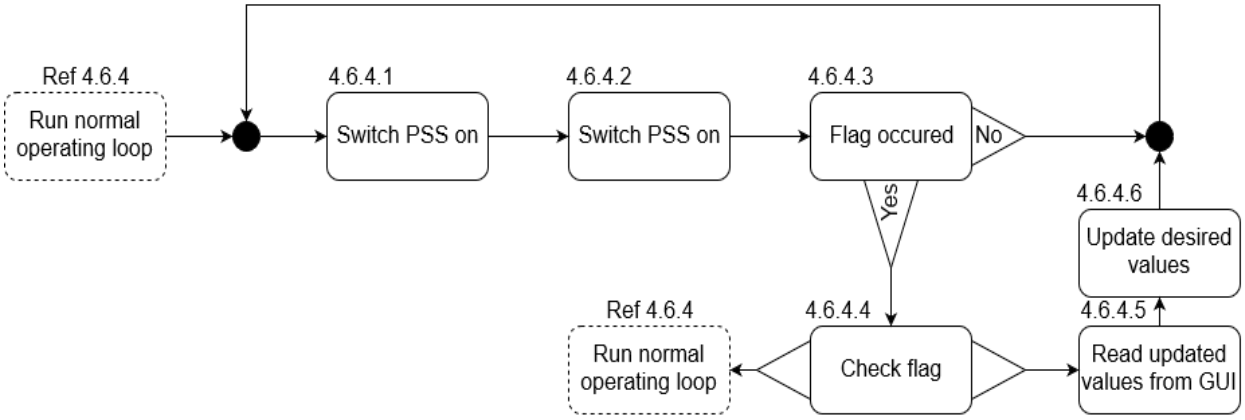


Figure 25: Normal output loop flow diagram

The next expansion that was done is done for *Run programmed output operating loop* (block 4.7.7). This block's expansion has a bit more work to it because a programmed output is implemented. The algorithm mentioned in *Run program output algorithm* is discussed and shown in Section 3.3.2. The expanded functional flow is seen in Figure 26.

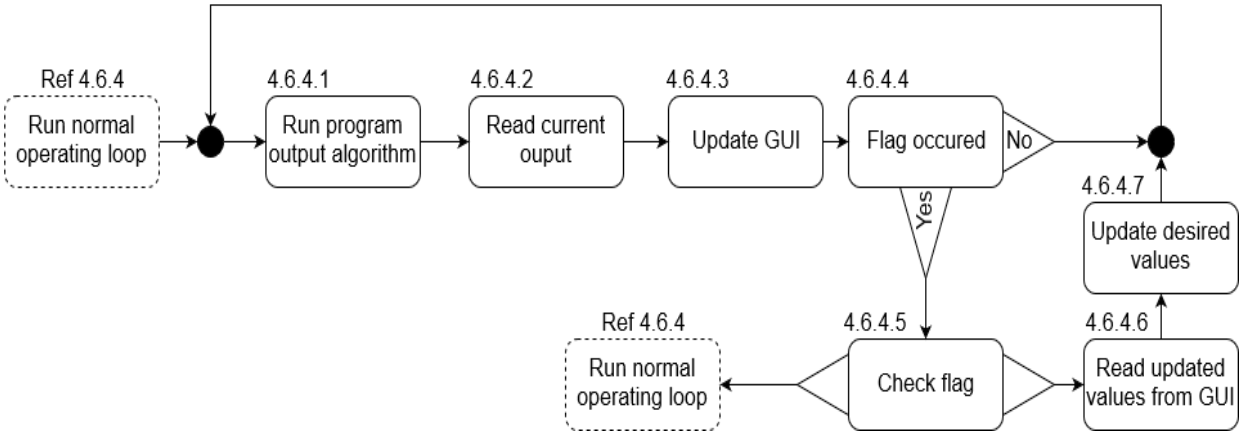


Figure 26: Programmed output loop flow diagram

These two expanded functional flow diagrams are at this point the only two that needs to be expanded.

3.3.2. Software

The software that is implemented is in control of two major parts, controlling the correct output levels via two DAC outputs and the displaying of the information retrieved from two ADC signals to the user via the GUI. Because two different processes have to run simultaneously, an operating system has to be implemented; this will give the option of running the controlling and the graphical displaying on different threads and thus running it seemingly simultaneously.

The software will also have to implement various algorithms to control the various states that are implemented in the PSS. In Section 3.3.2.1 the operating system trade-off is done. In Section 3.3.2.2 the basic algorithms that are implemented are shown and discussed and finally in Section 3.3.2.3 the graphical user interface that is used is shown and discussed.

3.3.2.1. Operating system

The operating system that is available to be implemented on the microcontroller we are implementing has been narrowed down to two possibilities, mbed or FreeRTOS.

mbed is an RTOS that is designed and developed by ARM the developers of the MCU used on STM developer boards. This means that the compatibility between mbed and the STM microcontrollers is brilliant. The microcontroller being implemented (discussed in Section 3.3.3.3) also comes with mbed enabled.

FreeRTOS is a freeware RTOS with a lot of tutorials and explanations online that will come in handy when trying to implement it. It also has been used in numerous different project to achieve the desired results I am after. VisualGFX the framework that is used to implement the graphical interface of the GUI is developed as a FreeRTOS task, thus if FreeRTOS is implemented integration is straightforward.

Thus taking these points into consideration, the real-time operating system that is implemented is FreeRTOS.

3.3.2.2. Algorithms

Currently, there is only one algorithm that is developed before the implementation phase starts, will the implementation of the code is occurring more algorithms are sure to come up, and when this happens the algorithms are developed and implemented. The algorithm that is already known about currently is shown in Figure 27.

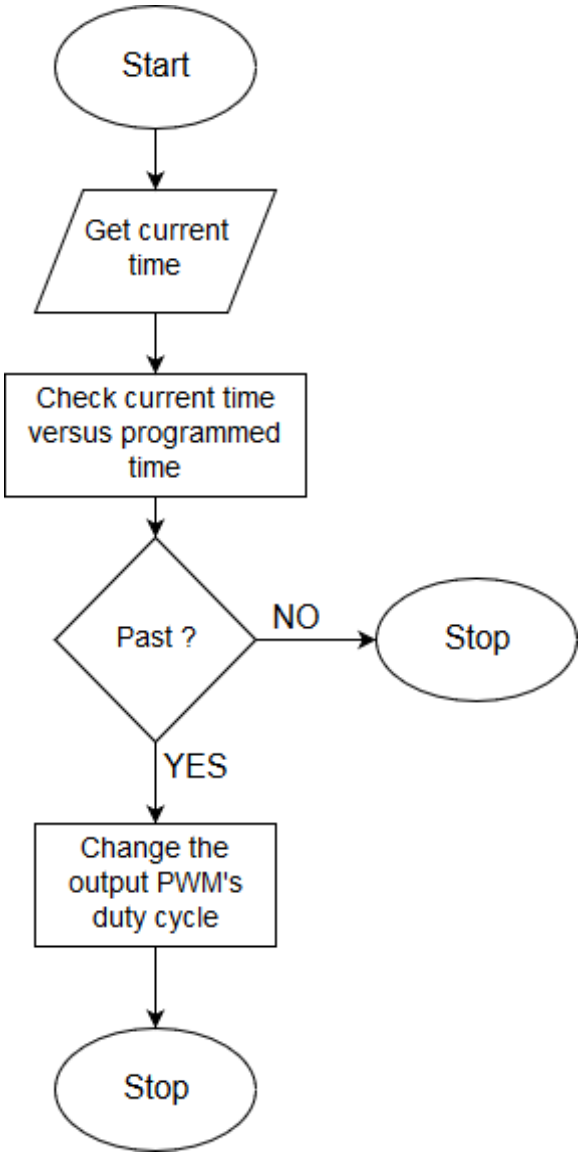


Figure 27: Programmable output algorithm

3.3.2.3. Graphical user interface

The GUI is implemented on a 4.3” TFT LCD (Section 3.3.3.3). The GUI will need to comply with the following requirements,

- Have the user able to input the desired voltage and current max
- Show the user the current voltage and current
- Allow the user to choose to implement a programmed output
- Have the main output on button

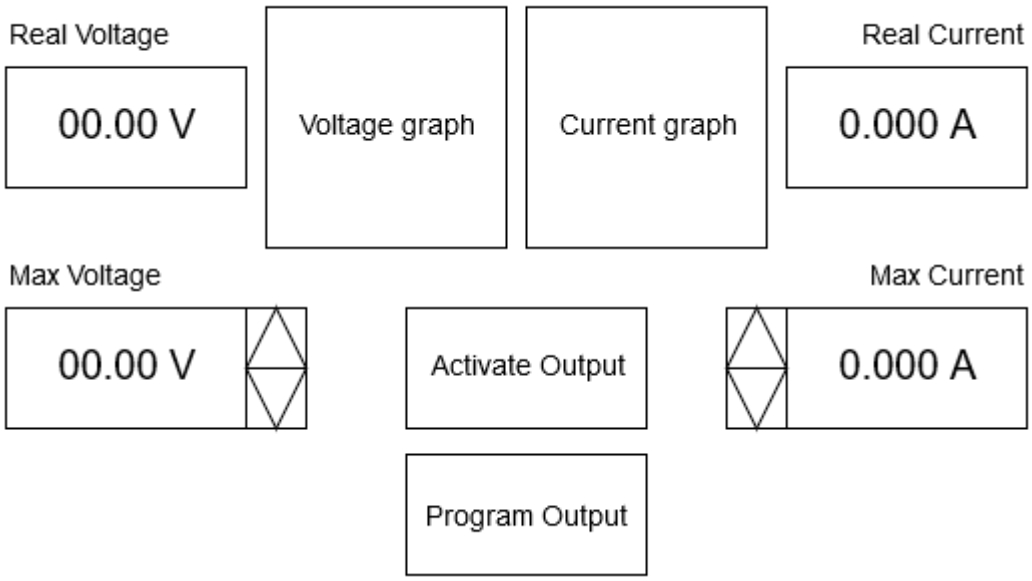


Figure 28: Mock-up of the GUI

In Figure 28 a mock-up of the GUI that is going to be implemented is seen. This GUI adheres to all the requirements and displays them in a manner that is easily readable and intractable on the 4.3" LCD.

The visual aspect of the GUI is done by implementing the VisualGFX framework; this framework was developed specifically to be implemented on embedded systems, by providing low memory usage with fast rendering speeds and visual aspects that are both pleasing to the eye and easy to interact with.

3.3.3. Trade-offs

For the architecture shown in Figure 24, there are a couple of trade-offs that should occur; these trade-offs are shown below in Sections 3.3.3.1 to 3.3.3.6.

3.3.3.1. Control Circuitry

The control circuitry that is implemented is a voltage regulator circuitry; this is the centre point of the system. This controls the output levels of the PSS and receives the PWM signal from the microcontroller. There are two approaches that are taken to implement the circuitry,

- Discrete;
- Integrated.

What is meant by a discrete controller is that the whole circuit is built by the user, this means that the stability has to be corrected by the user however it is built to sustain larger amounts of power dissipations compared to integrated circuits and is very cheap to build.

On the other hand is integrated chips, these chips are designed and built by manufacturers and thus meaning that your power dissipation capabilities are limited to the specifications of the chosen chip. However, the stability issues are sorted by the manufacturers and thus not the user's problem.

Due to the stability issues of discrete systems the use of an integrated voltage regulator chip is chosen. The specifications of this chip should be

- 0-3A rating;
- 0-30V rating;
- 90W rating.

3.3.3.2. Graphical user interface

From Chapter 2 Section 2.6.3, it was narrowed down to two options using a computer interface or an LCD. In that section, there are also given information regarding both methods with the up and downsides of both of them lifted out.

However taking into consideration the specifications of this project the User interface was chosen as an LCD.

Because this LCD is used for user input as well, desired voltage and current limits and if a pre-set programmed output should be used, it is clear that this LCD should have touch screen capabilities.

3.3.3.3. Microcontrollers

In Chapter 2 a comprehensive look at the microcontrollers are given, other microcontroller brands were left out due to the familiarity of the STM's layouts, capabilities and coding. The fact that the microcontroller will only have to implement two DAC signals to control the output maximums of the PSS and don't have to control the whole circuitry the load that is on the microcontroller and more importantly the speed at which the microcontroller has to run has also been reduced by the implementation of controlling circuitry.

This opens the window to use a microcontroller with better graphical display capabilities. Thus meaning that the STM32F746 becomes a very viable option for this implementation. The reason for this is,

- It has a speed capability of 216 MHz;
- 1Mbyte flash memory;
- It has three 12 bit ADC;
- Two DACs available;
- It has a built-in 4.3” TFT LCD.

However, there are other microcontrollers to look at also, however taking into consideration that a touchscreen LCD is needed for the Graphical user interface and that 2 DACs are needed to control the current and voltage levels of the PSS. The best part of implementing this microcontroller is that no extra funds have to be spent on buying one because one STM32F746 microcontroller is already available for use.

Taking all of these positives into consideration, it is obvious to see that the trade-off has been made and the chosen one is the STM32F739.

3.3.3.4. Housing

The housing options that are present in this implementation is,

- Steel;
- Perspex;
- Plastic;
- Aluminium.

Steel is eliminated due to the portability requirements of the project.

The remaining three options are compared in table format. The comparisons are seen below in Table 4

Table 4: Housing comparison

	Price	Secure	Weight	Difficulty
Steel	4	1	4	3
Perspex	3	2	2	3
Plastic	1	3	1	1
Aluminium	4	2	4	4

The comparison shows that aluminium is the more secure and shock resistant. However, it is more expensive than the other two options, and thus aluminium is eliminated. The other two options are a lot closer when comparing. With plastic being the easier option due to it being able to be fabricated in the NWU's fab labs, however, Perspex will give a better look to it and might be stronger than plastic.

Due to time constraints and the building capabilities of the student, a plastic housing is chosen as the option going to be implemented.

3.3.3.5. Cooling

The control circuitry will dissipate at maximum 90 W of power this together with the heat generated from the transformer will mean that some method of cooling within the housing is needed. The amount of heat that is generated by these components will not be major, thus concluding that ventilation holes in the housing together with a heatsink mounted on the voltage regulator are sufficient.

3.3.3.6. Connections

Due to the connection between electronics and a microcontroller there are a few extra connections to consider, all of the connections that are needed are,

3.3.3.6.1. DUT connection points

For the connection with the DUT, there are two main options, banana jack plugs or crocodile clamps. The commonly used connections are banana jack plugs this means that more users have the correct connectors for them and they are familiar meaning users will feel more at home using the power supply. Due to these reasons, banana jack plugs are chosen as the connection point for DUT's.

3.3.3.6.2. Connection point from wall socket

The connection with the wall socket is a male kettle cord connection; this was chosen due to familiarity and common availability of these plugs, thus meaning when a cord goes missing it can easily be replaced.

3.3.3.6.3. Connection from electronics to microcontroller

The connections between the electronics and microcontroller will need the implementation of a converter; this is because the electronics are in analogue and the microcontroller is in digital, thus meaning that the implementation of an ADC is needed. The electronics will run at a frequency of 50 Hz due to the input from the wall socket. This means that the ADC has to have the capabilities of sampling at a rate of 100 Hz due to Nyquist's theorem. The microcontroller that was chosen has 3 12 bit ADC available; this means that he are able to sample at a maximum sampling speed of 4096 Hz or samples per second thus meaning that the ADC on the microcontroller being implemented is more than sufficient.

3.3.3.6.4. Connection from microcontroller for electronics

As stated earlier the microcontroller will connect to the electronics via a digital to analogue controller. The microcontroller that was chosen in the previous section has 2 integrated DACs thus meaning that he have sufficient channels to use when setting the maximum voltage and current of our PSS.

3.3.3.6.5. Connection between LCD and the microcontroller

Because the microcontroller being implemented consists of an integrated LCD, there is no need to worry about the connections between the two parts.

3.3.4. Expanded Architecture

In Figure 29 the expanded architecture is seen. This architecture includes all the relevant communication channels and the relevant voltage levels of the different communication lines. This is the final architecture of the project and implemented all the needed function to meet the requirements of the project.

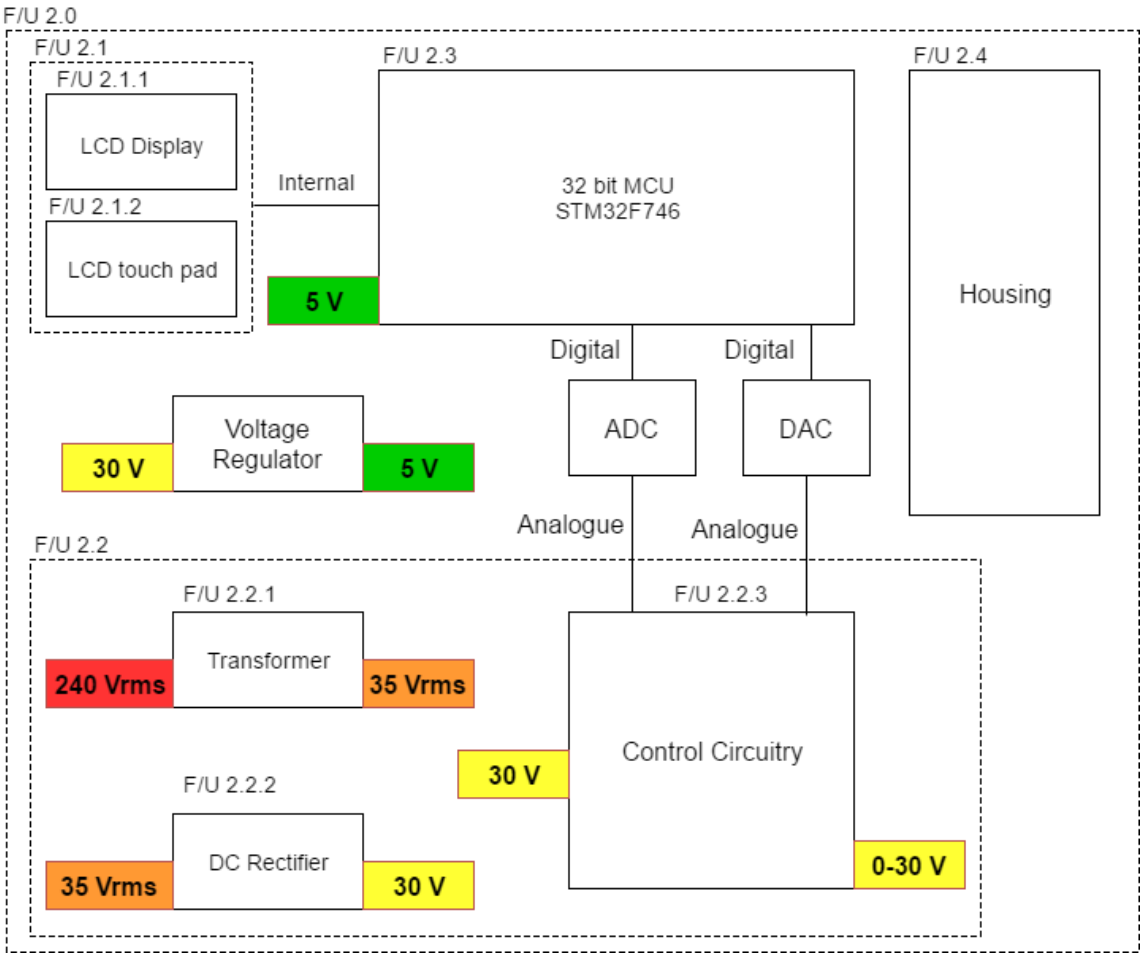


Figure 29: Expanded Architecture

The Control Circuitry block shown in Figure 29 is expanded in the figure shown in Figure 30, this is done to show the circuit that is implemented to achieve the controlling.

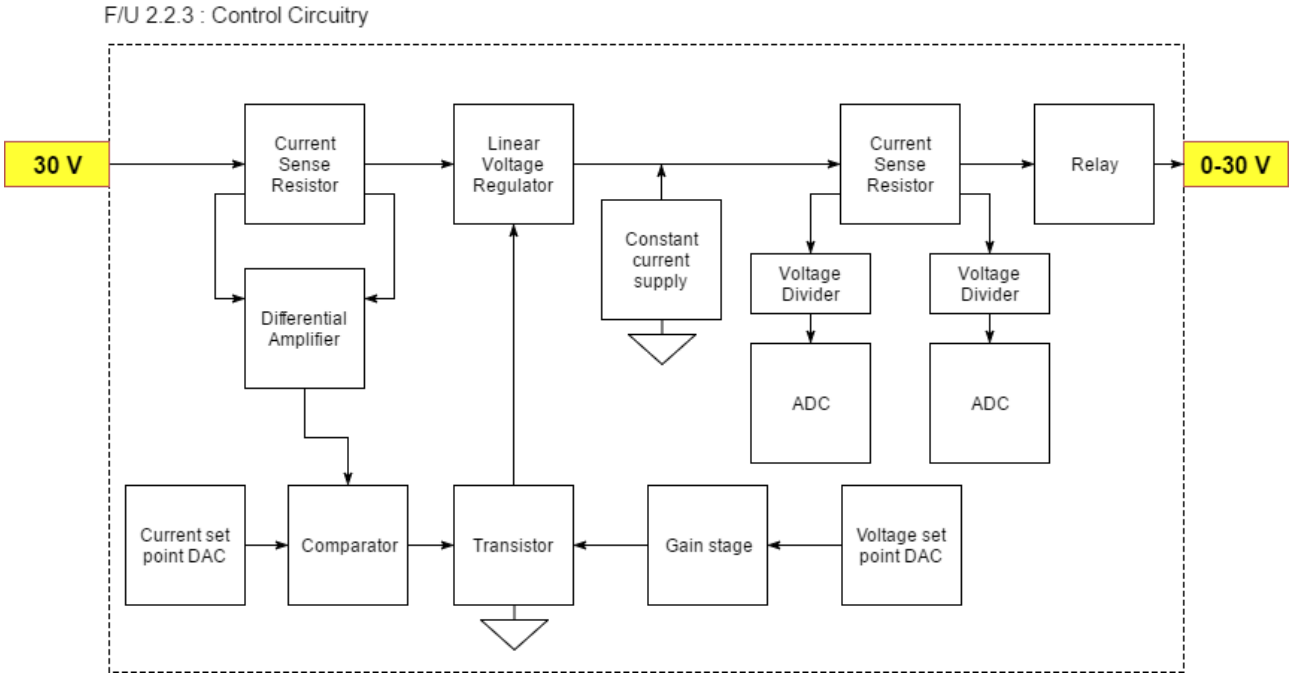


Figure 30: Control Circuitry Expanded

3.3.5. Expanded resource allocation

The functional units shown in Figure 29 together with the functions shown in the operational flow diagram in Figure 18 are used to develop the expanded resource allocation. The expanded resource allocation is seen in Table 5.

Table 5: Expanded resource allocation

	Unit 2.1.1 LCD Display	Unit 2.1.2 LCD Touch Pad	Unit 2.2.1 Transformer	Unit 2.2.2 DC Rectifier	Unit 2.2.3 Control Circuitry	Unit 2.3 Microcontroller	Unit 2.4 Housing
4.1 Setup PSS							✖
4.2 Switch PSS on		✖				✖	
4.3 Connect DUT					✖		
4.4 Configure Output		✖				✖	
4.5 Configure Program		✖				✖	
4.6 Activate Output		✖	✖	✖	✖	✖	
4.7 Output Active	✖		✖	✖	✖		
4.8 Deactive Output		✖	✖	✖	✖	✖	
4.9 Disconnect DUT					✖		
4.10 Switch PSS off		✖				✖	
4.11 Pack up PSS							✖

From Table 5 it is seen that the intended solution to the project is viable. This shows that a proper and thorough engineering design process was followed and that with the correct implementation this project is successful.

4. Implementation

4.1. Introduction

In this chapter, the research was done in Chapter 2 and the design done in Chapter 3 is implemented. This chapter will describe in detail the installation process of all programs used, configurations of STM board done in Keil uVision5, development of GUI in Visual Studios Community 2015, development of controlling algorithm in Keil uVision5, integration of the GUI, controlling algorithm and FreeRTOS in Keil uVision5 and finally the implementation of the electronics.

4.2. Software Installation

The platforms used for the implementation of the software aspect was,

- Visual Studios Community 2015, Version 14.0.25431.01;
- Keil MDK-Lite uVision Version 5.2;
- TouchGFX Framework Version 4.8.0;
- STM32 ST-LINK Utility Version 3.9.0.

Visual studios are downloaded from Microsoft's website and STM32 ST-LINK Utility from ST's website and be installed normally following the installer guide. Due to the availability of the professional version of Keil MDK the Lite version was used. This version is downloaded from ARM's website. The Lite version of Keil MDK imposes a code size limit when compiling. This meant that the compiling of the program had to be done by using TouchGFX MingW Environment, this process is discussed in further detail later on in this chapter.

The TouchGFX Environment was downloaded from their website and installed in a root directory. This is needed due to the MinGW compiler, allowing no spaces in file names. The GUI was developed from the empty template that is provided in the install directory of TouchGFX.

4.3. Configurations

The configuration of the software is divided into two main parts, the configuration of the environment and the start-up code for the STM32F746-Discovery board.

4.3.1. Environment

The empty template folder is moved to a new location, however, if this is done the correct adjustments should be made to the configurations files found under Application_Directory/config in both of the files found there the correct path to the Install_Directory/touchgfx should be added.

The MinGW compiler requires a makefile to work; the basic makefile is included in the project directory *Application_Directory/target/ST/STM32F746/gcc/Makefile*. This makefile is the bases around what the MinGW compiler works; then compiler errors will occur.

All peripherals that are needed for the project needs to be added to the makefile, to achieve this the board configuration files provided by TouchGFX needs to be altered. These file is found under *InstallationFolder/touchgfx/board/ST/STM32F746-DISCO/include/stm32f7xx_hal_conf.h* and *InstallationFolder/touchgfx/config/board/ST/STM32F746-DISCO/gcc/board.mk*. In the first file, the comments were removed before the appropriate peripherals that were needed, while in the second file the peripheral header file was included.

If any extra directories are added to the base project, which includes new project files, these directories should be added to the main makefile under *Application_Directory/target/ST/STM32F746/gcc/ Makefile*.

4.3.2. STM32F746-Discovery start-up code

Normally when using a microcontroller from ST, the start-up is automatically generated with the used of STM32 CubeMX. However, due to the TouchGFX having compatibility issues with STM32 CubeMX for the specific microcontroller in this project, this method could not be used.

This meant that the configuration code of the project was manually done, in a separate file *PSS_Config*. This file included all the necessary peripheral configurations that need to be done on start-up together with the MSP code. The needed start-up code was developed by using the HAL library documents [20] and by using STM32 CubeMX for example code [21].

4.4. Firmware

In this section, the firmware that was developed is discussed. The section is broken into different parts that address one specific aspect.

4.4.1. GUI

The GUI was developed by implementing the TouchGFX framework, as briefly said earlier. This framework is a total revamp of the current graphics portion of ST's library. It is easy to develop, has great templates and examples to work from, has very fast rendering speeds, low memory usage and the whole GUI is integrated with FreeRTOS [22].

TouchGFX was also picked to use due to its drag and drop capabilities when designing GUIs. This meant that by using their designer the GUI could be easily changed the layout could easily be altered and the code regenerated.

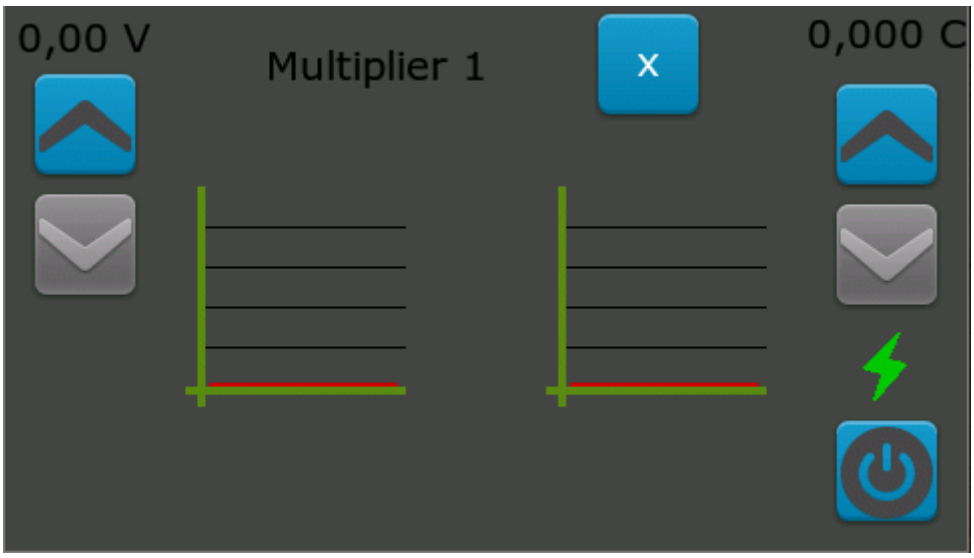


Figure 31: Finale GUI

The GUI was developed in Visual Studios [23] using C++ as the language. This language and IDE was chosen due to the TouchGFX framework. In Visual Studios, TouchGFX provides the feature of having a real-time debug for the GUI. This means that if debug is run in Visual Studios the GUI is shown and you can test all the features in real time without having to flash to the Developers board first. This decreased the time it took to develop and debug the GUI greatly.

The GUI was developed with the main thought being user friendliness and pleasing design. Using the design created in Section 3.3.2.3, the final GUI was designed and implemented. This design is shown in Figure 31. As is seen from this picture the finale design changed from the mock-up shown in Figure 28. This was due to the actual size of the GUI. The GUI was firstly designed exactly like the one shown in the mock-up. However, the graphs became illegible, and the design was altered around that fact.

4.4.2. Controlling Algorithm

The controlling algorithm was designed in Keil MDK-Lite uVision 5 [24] implementing the C language and the HAL framework. These environments and languages were chosen based on experience and preferences.

The controlling algorithm has 5 main tasks that need to be completed per cycle or within a set amount of time. These four tasks are:

- Checking GUI task Queues and Semaphores for new data;
- Updating the peripherals if new data is received;
- Checking the analogue to digital converters for new data;

- Sending the new data to the GUI via Queues and Semaphores;
- Giving the GUI process time to catch user inputs.

When having a look at all these tasks, there is no task that has critical importance and will break the system if not done within a set amount of time. This is due to the controlling circuitry that was implemented. The only constraints are that the GUI be checked for User Input every 200 ms, new data from GUI checked after user input has been checked for and that the analogue to digital converters asked for new data at least once every 500 ms.

These constraints are rather loose and open up the opportunity for various methods of controlling being implemented. The method used is to implement two different tasks. The GUI task and the Controller task. The Controller task has control over the processor most of the time and gives the GUI 20 ms of time to run all its functions and check for user input every time one of the other tasks have been completed.

This means that, let's look at an example of how the program would run. Let's say each of the 4 mentioned tasks (excluding the GUI) take about 5 ms to complete (this is much longer than it would actually take but this was just done as an example) this means that every 5 ms the GUI gets 20 ms to check for user input and do all its functions. The four tasks will also run at least once every 95 ms. This means that all the tasks are running faster than what is needed and the GUI has a refresh rate of 40 Hz.

Another benefit of the fast speed is that the ADC values are summed together for 500 ms, and then the average is calculated and sent to the GUI; this means that any spikes are irregular data being read would be removed.

4.4.3 Real-time operating system

The Real-time operating system that is implemented is FreeRTOS. This was due to the TouchGFX framework being already integrated with FreeRTOS and built on a single task. FreeRTOS was the main integration part between the controlling algorithm and the GUI.

The integration is done by having two tasks that worked alongside each other. The data transfer between the tasks was handled by Queues and Semaphores. By the implementation of this system, the data transfer between the TouchGFX framework and the ST HAL framework is made easy, and no problems are occurred to transfer data between C and C++ processes.

The RTOS also increased the responsiveness of the GUI and kept the controlling software running smoothly in the background. All these factors chose to implement an RTOS and using FreeRTOS worthwhile and improved the user-friendliness of the project.

4.5. Electronics

In this section the components implemented to achieve the desired results for the electronic schematic shown in Figure 30 is discussed. After the components have been discussed the circuit is implemented in LTspice, and the circuit is shown to be working. The circuit is then implemented on a Veroboard to use as a proof of concept and prototype. This final design was then used to create a schematic and PCB layout in Altium [25].

4.5.1. Components

There are six major parts to the control circuitry, these are:

- Voltage regulator;
- Differential amplifier;
- Comparator;
- NPN bipolar transistor;
- Constant current supply;
- AC/DC Power supply.

As said in the design chapter all the components that were chosen for the final design is integrated chips, this removes instability problems and reduces the size of the resulting schematic.

In Section 3.3.3.1, the specifications of the desired voltage regulator together with some regulators that are used. The regulator that was chosen is the LT3080, to implement this regulator 3 of them is added in parallel, this will increase the price slightly, but it will reduce the heat generation of them significantly. The pinout of the LT3080 is shown in Figure 32 [26].

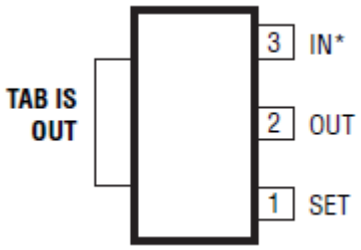


Figure 32: LT3080 Pinout

For the Differential amplifier and Comparator, an integrated solution was chosen, that is both cheap and easy to use. For the differential amplifier, the LT1991 was chosen, this chip is used with as a unity gain differential amplifier and can swing to 40 mV of its rails. This meant that a negative and positive five-volt rail was needed. The pinout of the LT1991 is shown in Figure 33 [27].

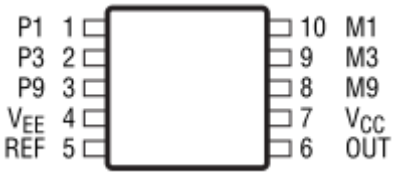


Figure 33: LT1991 Pinout

For the comparator the LM397 was chosen, this is a cheap and commonly used comparator that provides all the needed functionality and speed. The chosen chip can also work from a single supply thus meaning the five-volt rail required for the LT1991 will suffice to drive the comparator. The pinout of the LM397 is shown in Figure 34 [28].

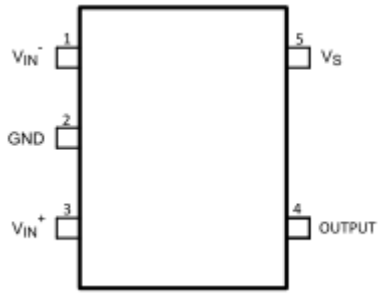


Figure 34: LM397 Pinout

For the NPN bipolar transistor, the common 2222 was chosen, this transistor was chosen due to its familiarity. The pinout of 2222 is shown below in Figure 35 [29].

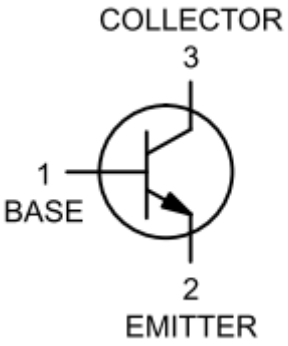


Figure 35: 2222 BJT Pinout

The LM134 was chosen as constant current supply; this is a 3 pin adjustable current supply that can easily generate the required 10 mA. The LM134 generates an amount of current directly correlated to the current that is flowing through a resistor connected between its set and bias pin. This resistor was calculated as 133 Ω. The pinout of the LM134 is shown in Figure 36 [30].

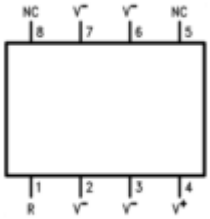


Figure 36: LM134 Pinout

In Chapter 3, the AC/DC power supply was discussed, and the thought then was to implement a discrete AC/DC power supply by implementing a transformer and rectifier bridge. However, after some extra designing and research has been done the conclusion has been made to change the discrete AC/DC power supply to an integrated switch mode power supply. This will increase the stability of the system, reduce the designed strain. This will increase the price of the final design. However, if the price at a later stage wants to be reduced, the SMPS is replaced by a self-designed power supply. The SMPS that was chosen to be implemented is the ABC120 a photo of this SMPS is seen in Figure 37 [31].

Figure 37: Switch mode power supply

Next, a 24 VDC relay was chosen, this was later discovered to be the wrong relay due to the coil switching voltage being too high. However, by implementing a gain stage, the correct coil voltage was achieved, and this corrected the situation. The relay that was implemented was an IM Relay; the pinout is shown in Figure 38 [32].

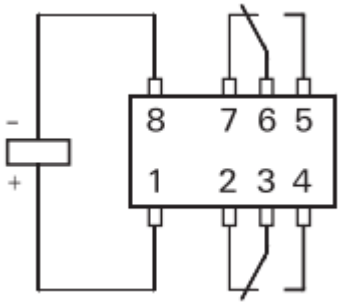


Figure 38: Relay pinout

The final components that were needed are linear regulators to create the positive and negative five-volt rails. The regulators picked are the L7805 and L7905. They were chosen due to familiarity, and common availability. The pinout of both these devices are shown in Figure 39 and Figure 40 [33][34].

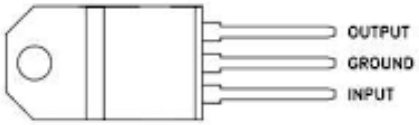


Figure 39: L7805 Pinout

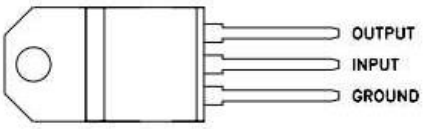


Figure 40: L7905 Pinout

The components listed above were chosen for the final implementation on a Printed Circuit Board (PCB). These components were chosen due to the familiarity of the product, being integrated chips, cost and being easily obtainable.

4.5.2. LTspice

In this section, the design shown in Chapter 3 Section 3.3.4 is implemented in LTspice [35]. This was done to provide a proof of concept of the chosen design. In Figure 41 the circuit used for simulations are shown. This circuit includes the models of all the components that are implemented.

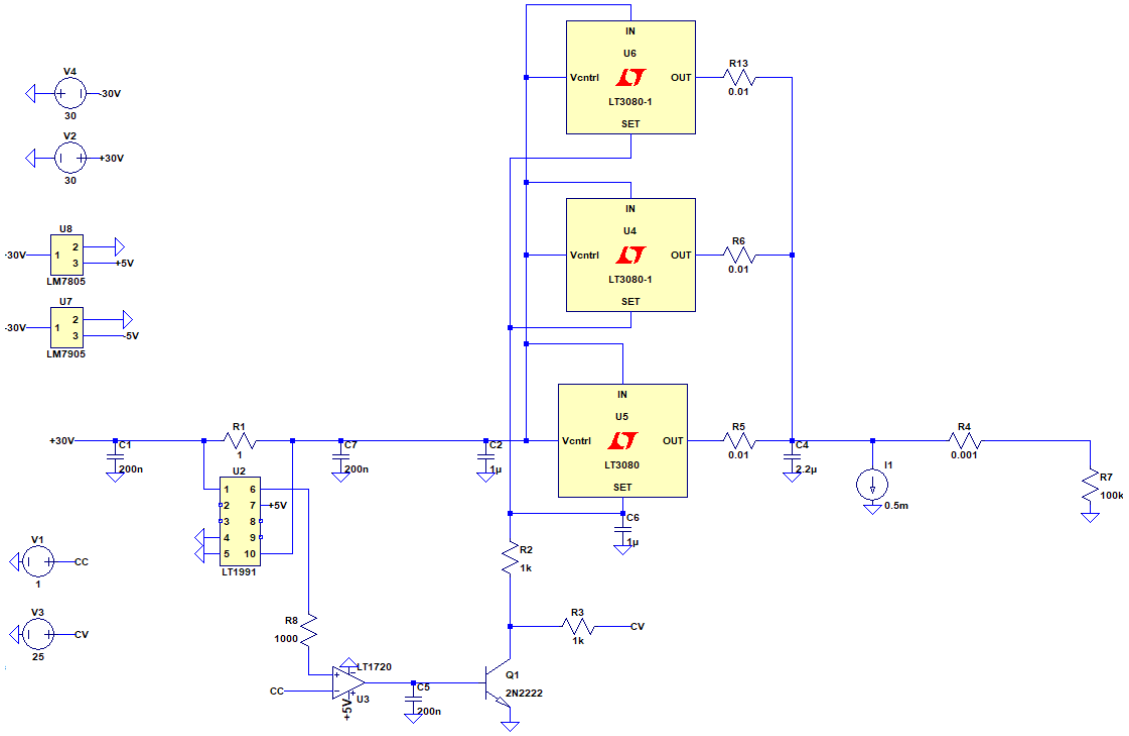


Figure 41: Simulated Circuit of the main circuit

This circuit was simulated for constant current, and constant voltage situations, the current limiting or constant current simulations is seen in Figure 42. From the results, it is seen that even though the voltage limit has been set for 25 V (as indicated by the blue line), the output voltage is limited to under 1 V (shown by the red line) this is due to the current limit being set to 10 mA. This proves that the simulated circuit does present constant current or current limiting capabilities.

In Figure 43 the circuit was simulated where the current limit was changed to 1 A. This was done to show the constant voltage capabilities of the circuit. As is seen from the graph the voltage limit is set to 25 V (as indicated by the blue line) and the steady-state output of the circuit (shown by the red line) is very close to 25 V. This proves the constant voltage capabilities of the circuit.

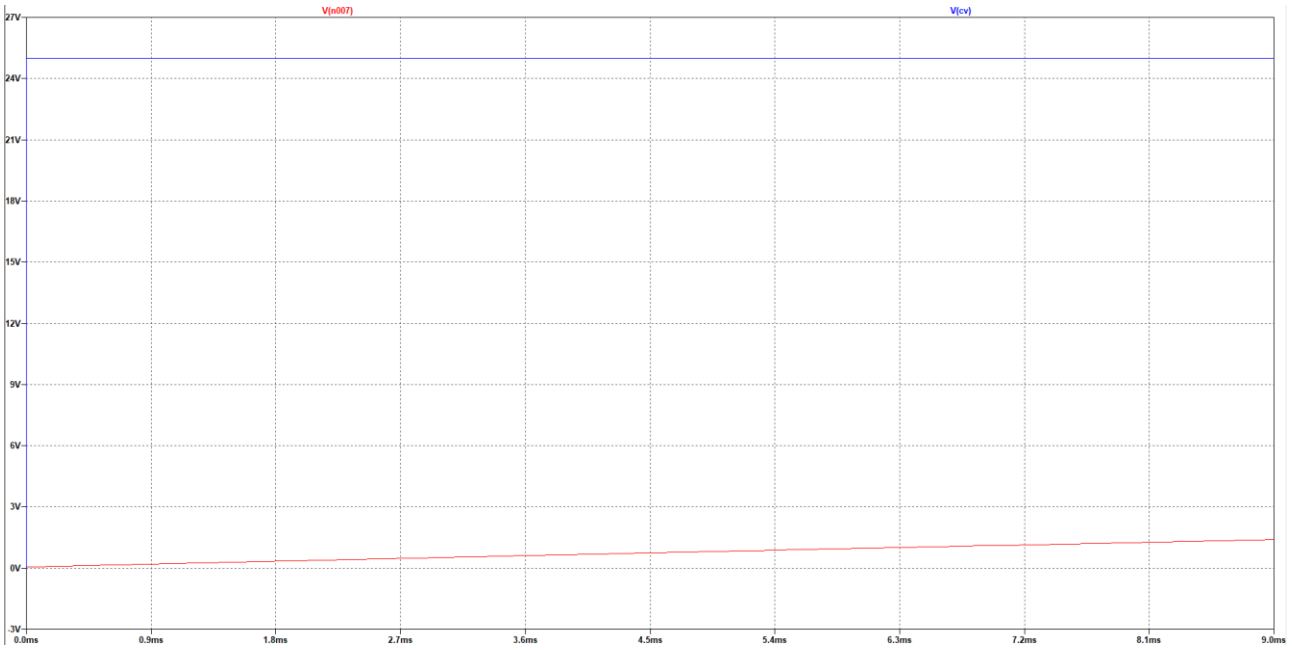


Figure 42: Constant Current of the main circuit

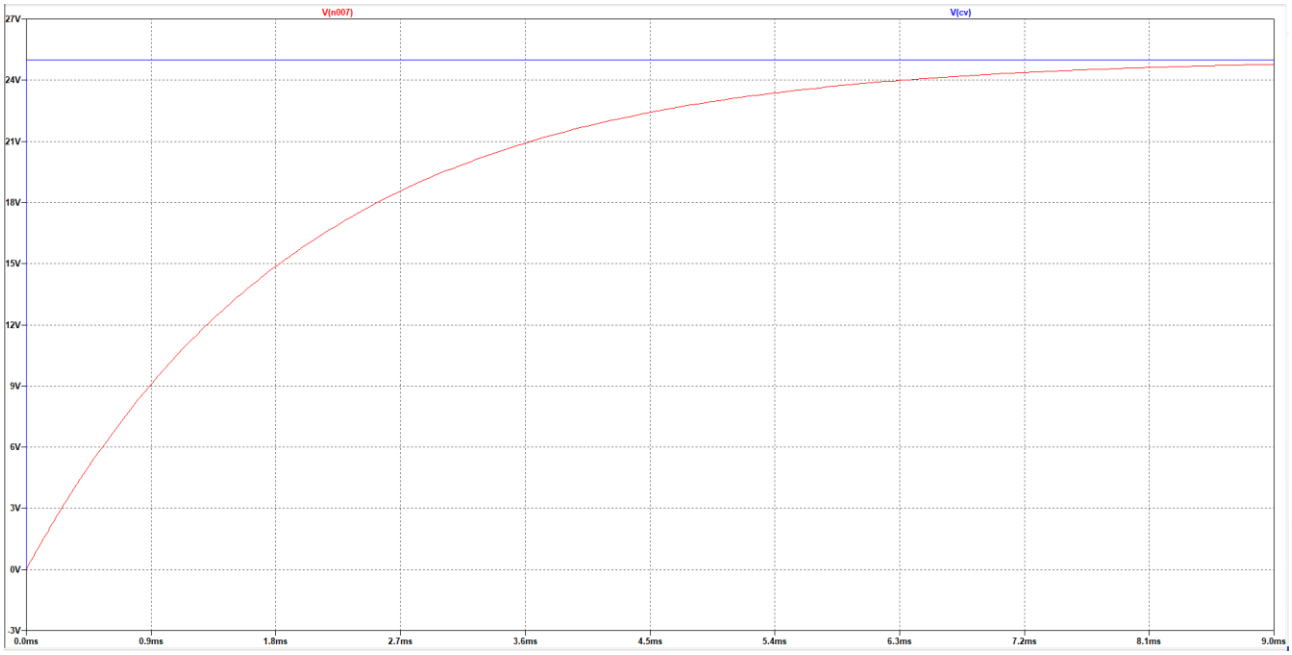


Figure 43: Constant Voltage of the main circuit

For the next simulations, the models of the actual components were removed, and operational amplifier equivalent circuit are added the LT3080 regulator was also replaced with only 1 due to this only being a proof of concept and prototype, this meant that the current limit of the circuit is reduced to 1A. This is done to simulate the circuit that is built on the Veroboard.

The reason behind different components being used for the breadboard circuit and prototype is due to availability. All the components being used for these builds could be sourced from local stores, and thus the concept could be proven without waiting for components to arrive at online stores. The

chips that are chosen to be implemented on the PCB are identical to the operational amplifier circuits they are just integrated.

The new simulated circuit is shown in Figure 44.

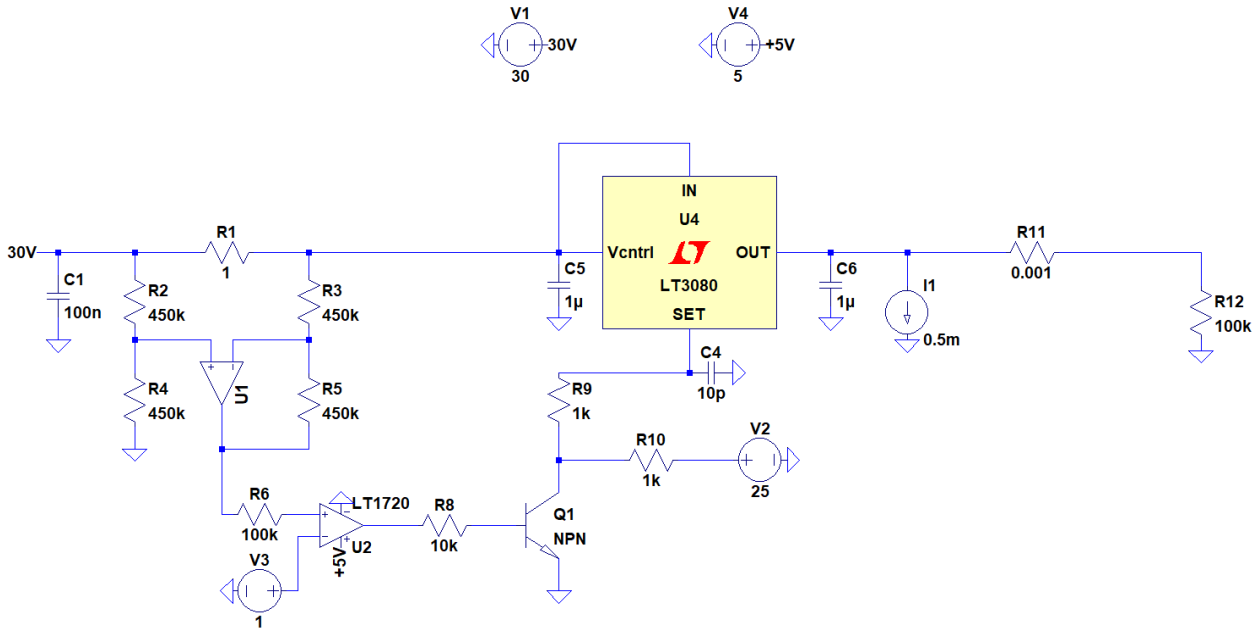


Figure 44: Simulated circuit with Operation Amplifier circuits

In Figure 45 the new simulation was run with the current being limited to 10 mA, as is seen just like the first simulations the results are the same. The output voltage (shown with the red line) is limited to less than 1 V, even though the set voltage 25 V (shown by the blue line). This proves that the new circuit still exhibits current limiting.

The simulation was run again. However this time the current limit was increased to 1 A., The results are shown in Figure 46. As is seen from the red line, the new output voltage is now 25 V, this is the same as the set voltage indicated by the blue line. The only difference that is seen between the new simulations and the simulation shown in Figure 43 is that the transient response has been reduced significantly. These results are due to the ideal Op Amps being implemented.

Due to some complications with implementing the Digital to Analogue converters together with the LCD on the developer board, the onboard DAC had to be replaced by Pulse Width Modulation signals for the prototype board; external DAC's is used for the final implementation.

The simulations were repeated with the changes being made to the circuit. The new set points of the system were made by implementing PWM signals filtered using low pass filters. This process will impose some setting point limitations and some ripple.

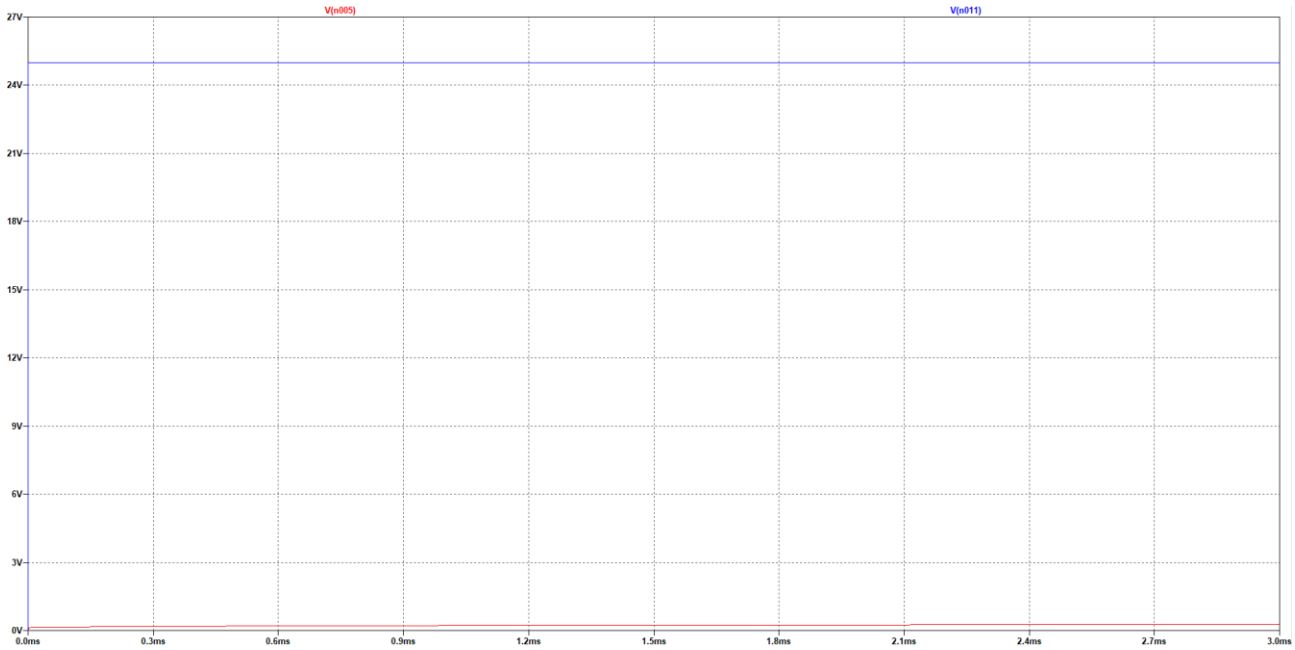


Figure 45: Constant Current of Op Amp circuit

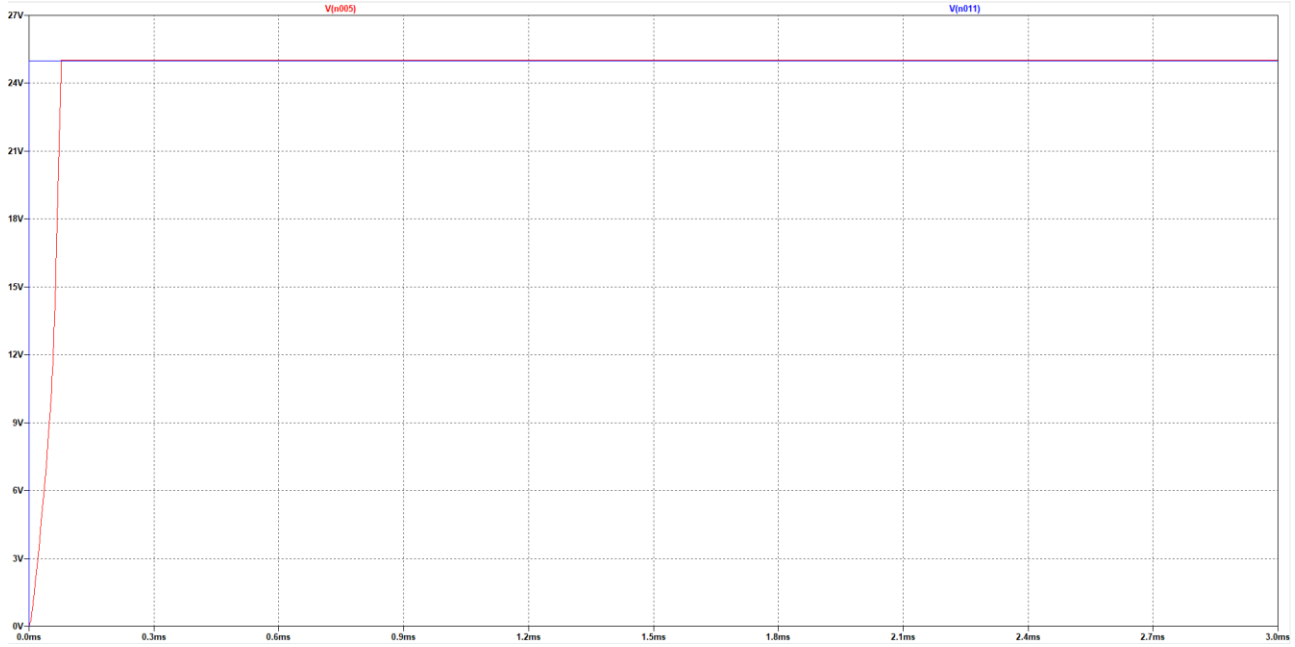


Figure 46: Constant Voltage of Op Amp circuit

The new circuit is shown in Figure 47. This circuit contains the low pass filters and PWM signals represented by voltage sources producing a pulse signal. In Figure 48 the PWM signal filtering is proven. From the graph, it is seen that a ripple voltage is introduced however due to this not being the final implementation this ripple is ignored.

The DC voltage of a PWM signal is set by altering the duty cycle of the PWM signal. This means that the set point fineness is limited by die timer speed of the PWM signal on the developer board. This meant that the set point fineness for the prototype board was reduced to 6 mA and 60 mV.

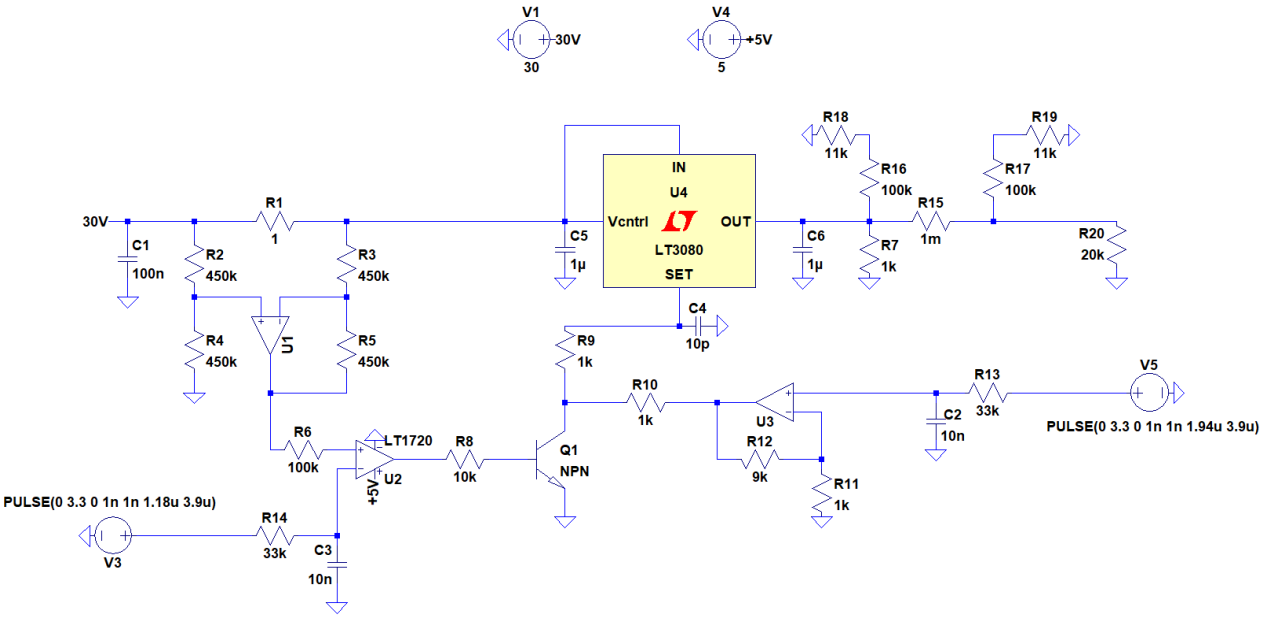


Figure 47: Simulated circuit with PWM

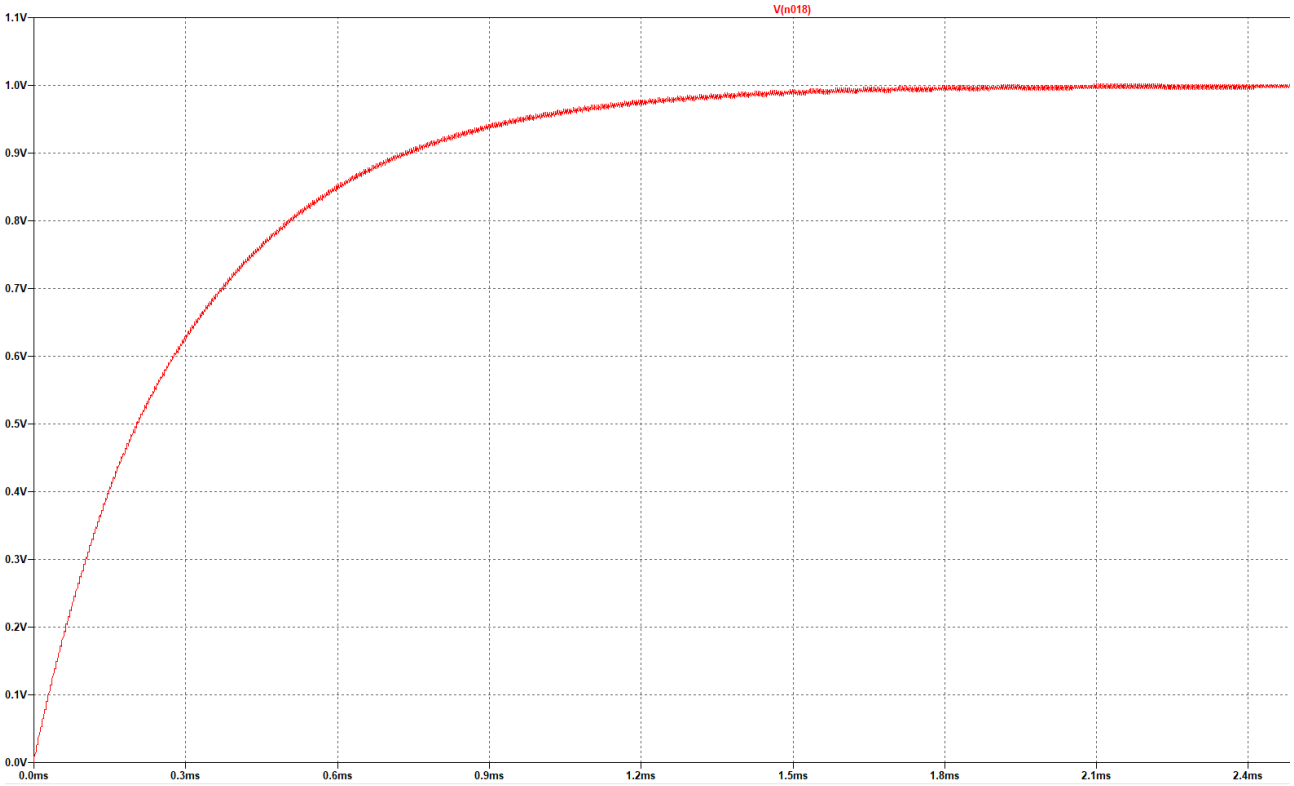


Figure 48: PWM signal to DC signal

For the voltage set point, the PWM output voltage had to be increased by a factor of 10; this is due to the developer board being limited to an output voltage of 3.3V. To achieve this again stage is implemented. This can also be seen on the circuit in Figure 47.

On the following page in Figure 49 and Figure 50, the simulation results are seen, these simulations were done under the same conditions as the previous two simulations. As is seen from both graphs the results are the same as the previous editions.

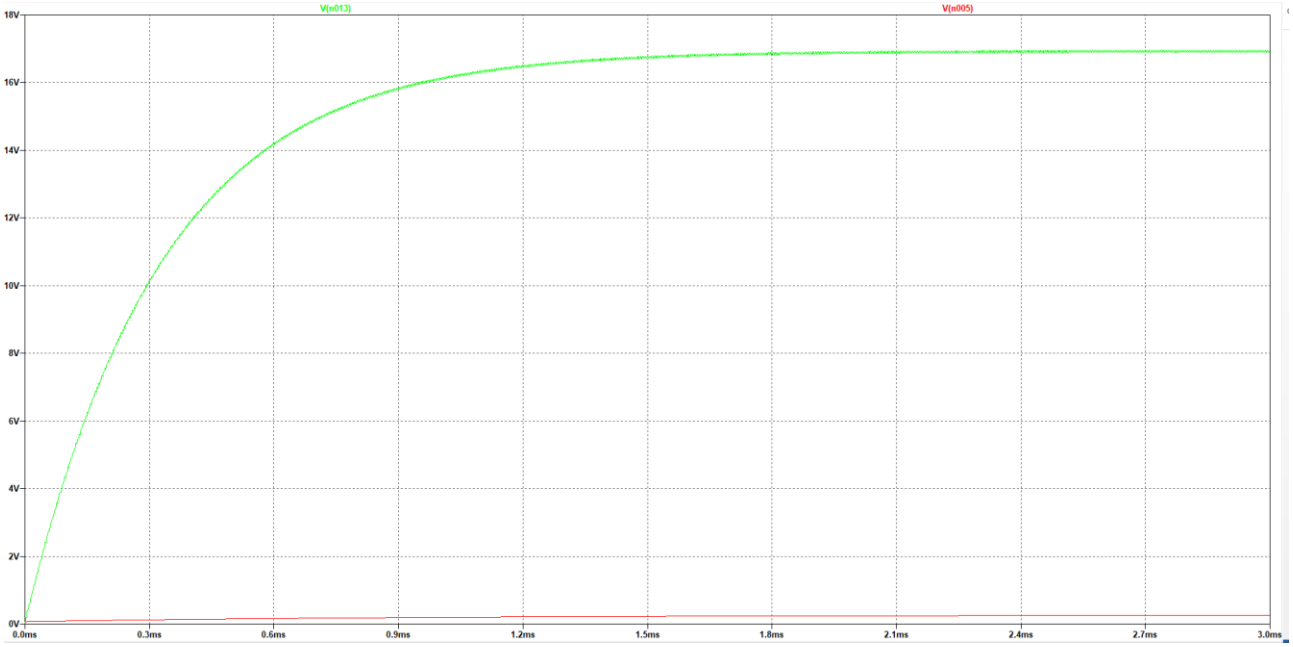


Figure 49: PWM circuit Constant Current simulation

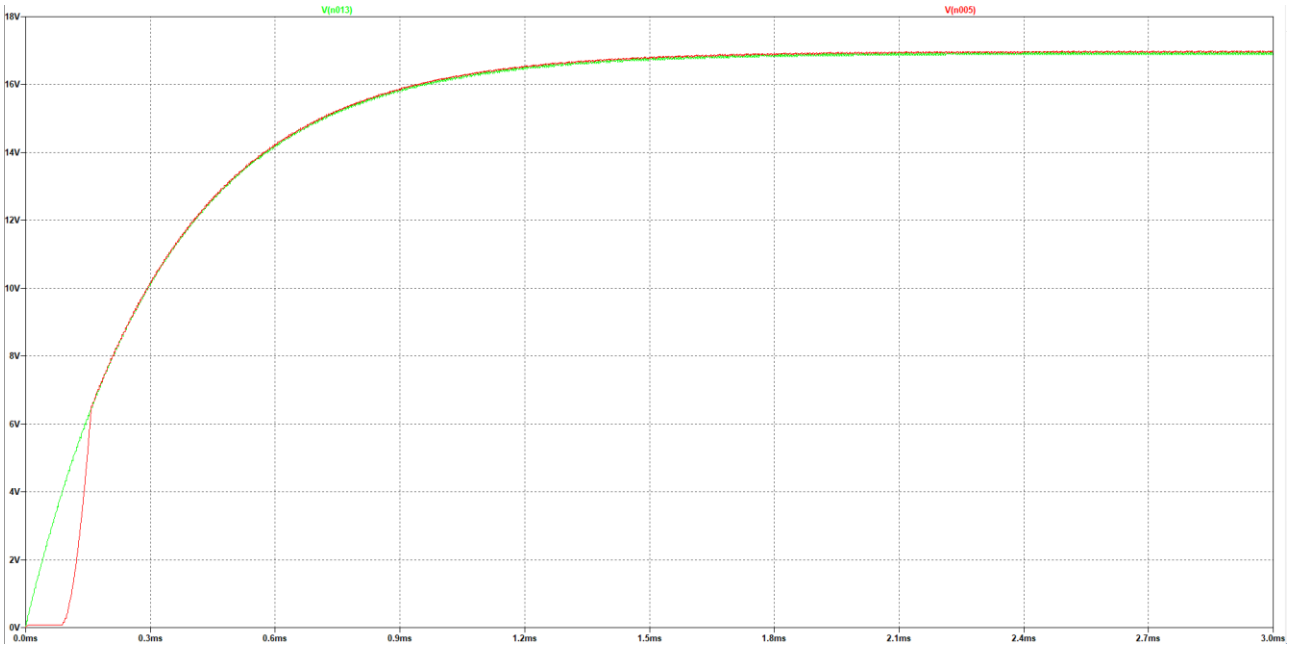


Figure 50: PWM circuit Constant Voltage simulation

This concludes the simulations done for the projects. The simulation results show that the concept correct is. This is the first step in proving that the final implementation is successful.

The next step is to build the prototype and see if the results are the same as what the simulations showed. This is done in the next section.

4.5.3. Veroboard implementation

To implement the circuit shown in Figure 47, LMV358 Operational Amplifiers was chosen. These Op Amps were chosen due to their availability in local shops, their familiarity and stability. The pinout of the package bough is shown in Figure 51 [36].

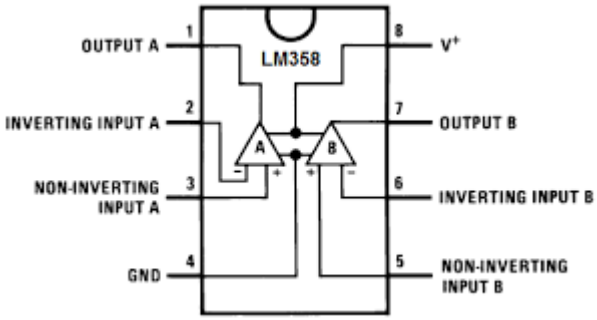


Figure 51: LM358 Pinout

The circuit was implemented and soldered onto a Veroboard. The prototype Veroboard is seen in Figure 52. This board was built modularly, meaning that each part was separately built and tested. After all, parts were built and tested the circuit was connected.

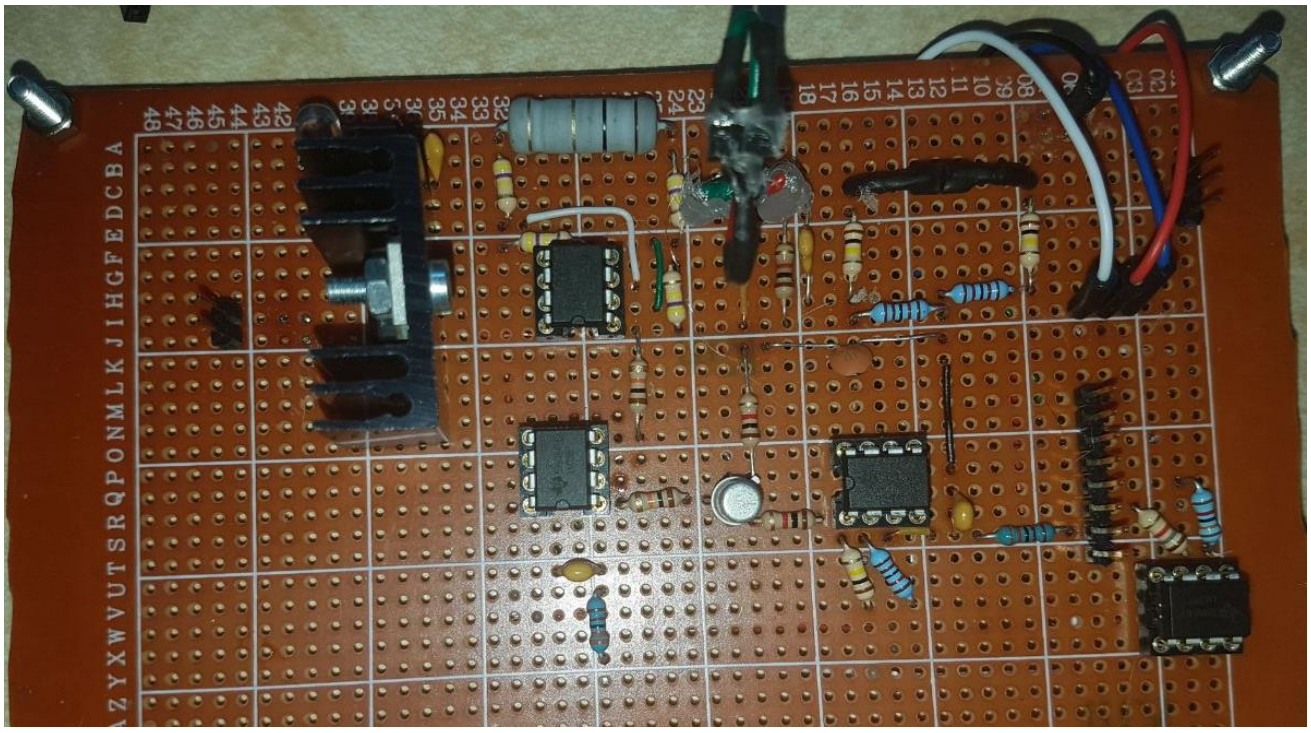


Figure 52: Prototype implementation on Veroboard

4.5.4. PCB Implementation

In this section, the process that was followed to go from the circuit used for simulation to the final populated PCB is handled. This process includes creating a schematic, then doing the PCB layout of the whole schematic, sending the layout to be printed and finally populating the PCB with the components discussed in Section 4.5.1.

The first step as mentioned is to create the schematic. This is done by using the circuit design shown in Figure 41 and Altium Designer [25]. The schematic has been divided into parts and done in separate files to keep clatter to a minimum. In Figure 53 the main controlling circuitry schematic is shown. As is seen various decoupling capacitors have been included to improve the noise generated by dying Switch Mode Power Supply.

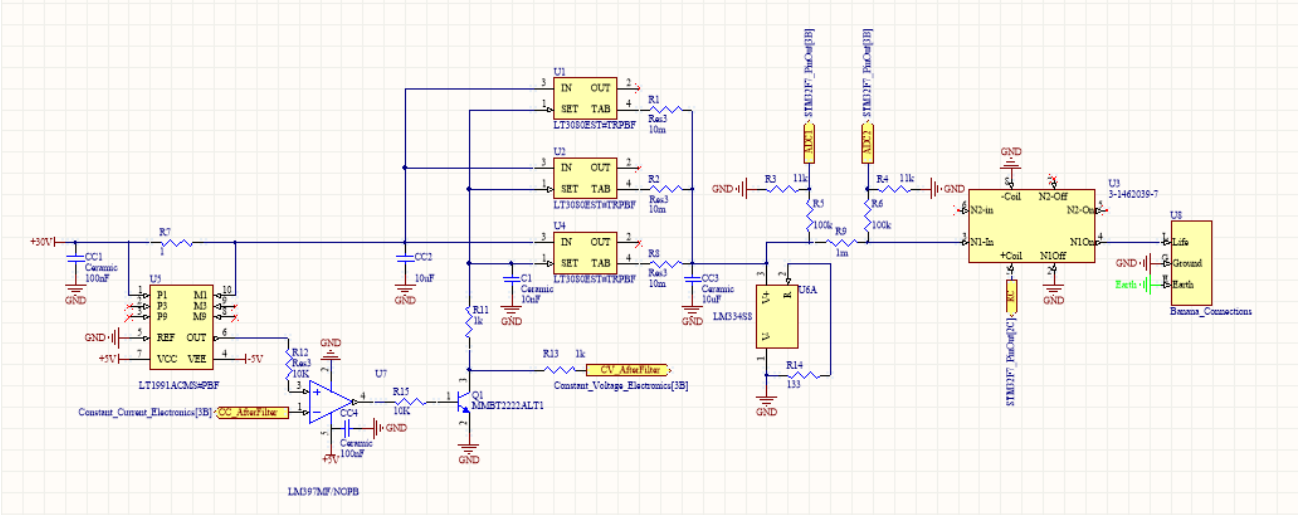


Figure 53: Controlling circuitry schematic

In Figure 54 the positive and negative five-volt rails are implemented. The circuit is designed to reduce any noise that might be present.

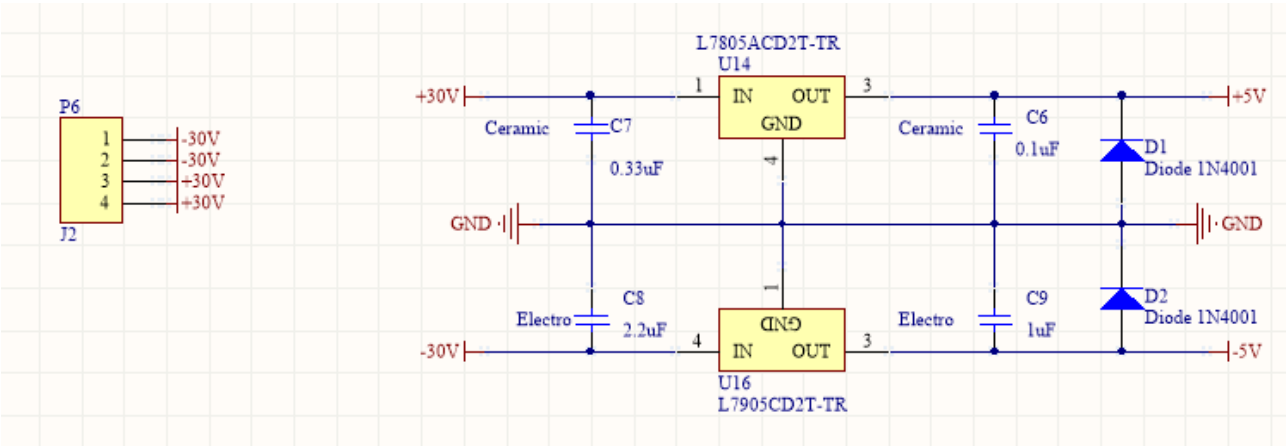


Figure 54: Rail voltage schematic

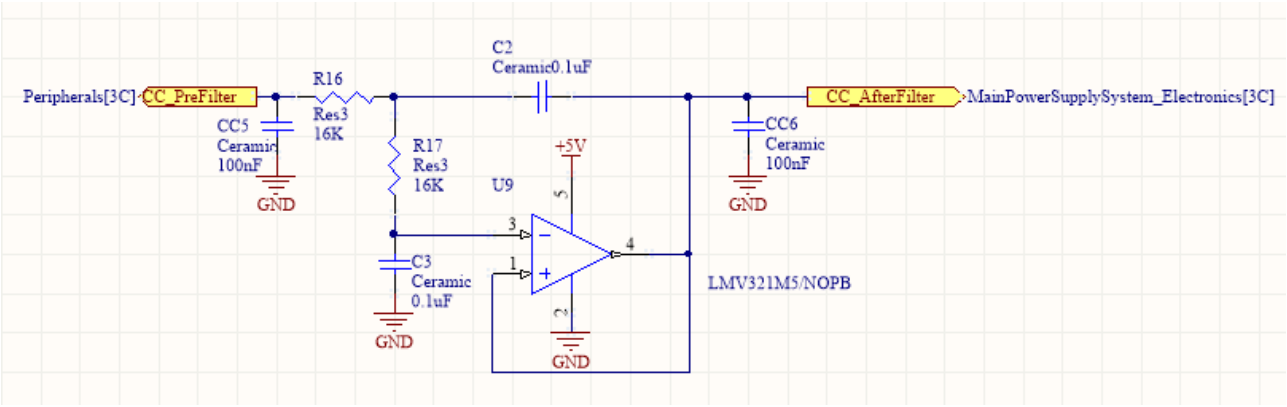


Figure 55: Constant current schematic

In Figure 55 the current limit setting DAC input is received, this input is firstly filtered by implementing a Sallen-key active filter. This is done to remove any noise that might be present after the conversion from digital to analogue has occurred.

In Figure 56 the voltage limit setting DAC input is received, and the same filtering method used to filter the current limit setting DAC is implemented. The extra stage present on the schematic is the gain stage that is used to multiply the filtered DAC output by 10.

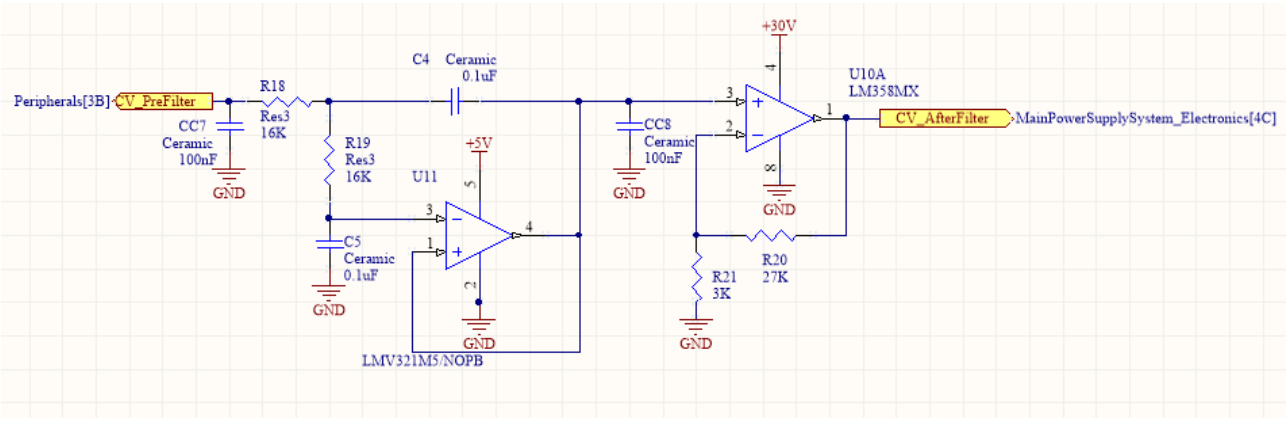


Figure 56: Constant voltage schematic

As said in Section 4.5.2. The onboard DACs could not be used due to the developer board implementing the same pins for controlling the LCD. This meant that external DACs had to be implemented. In Figure 57 the connections of the external DAC is seen.

The DACs that are implemented are Serial Peripheral Interface (SPI) enabled, and thus an SPI communication channel from the developer board is connected to both ADCs with the chip select (CS) pin being connected to two different General-purpose input/output (GPIO) pins of the developer board. The communication between the developer board and the DACs are done by first enabling the correct CS pin, then sending the data and then disabling the same CS pin again. This configuration means that only one SPI channel is used for two DACs

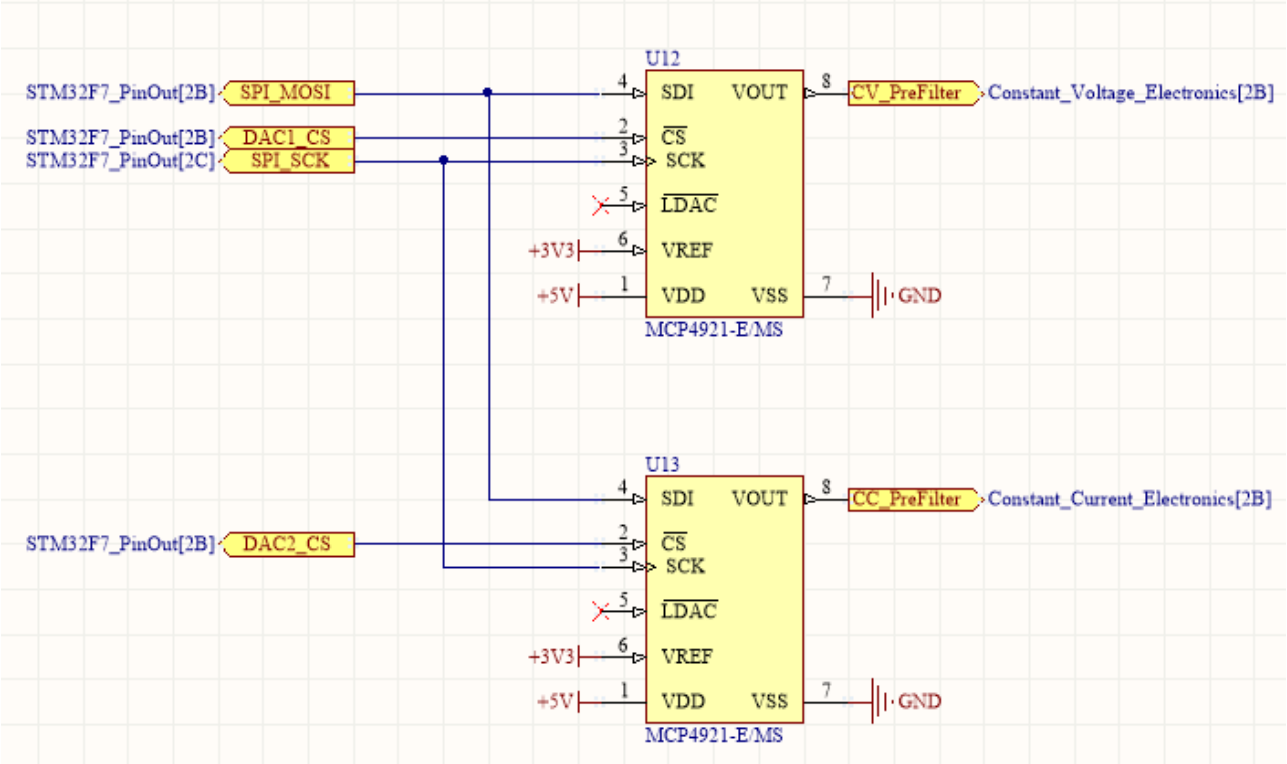


Figure 57: External DAC connection schematic

In Figure 58 the pin out of the STM32F7 is seen, this pinout has been created by using the user manual of the STM32F7 [37].

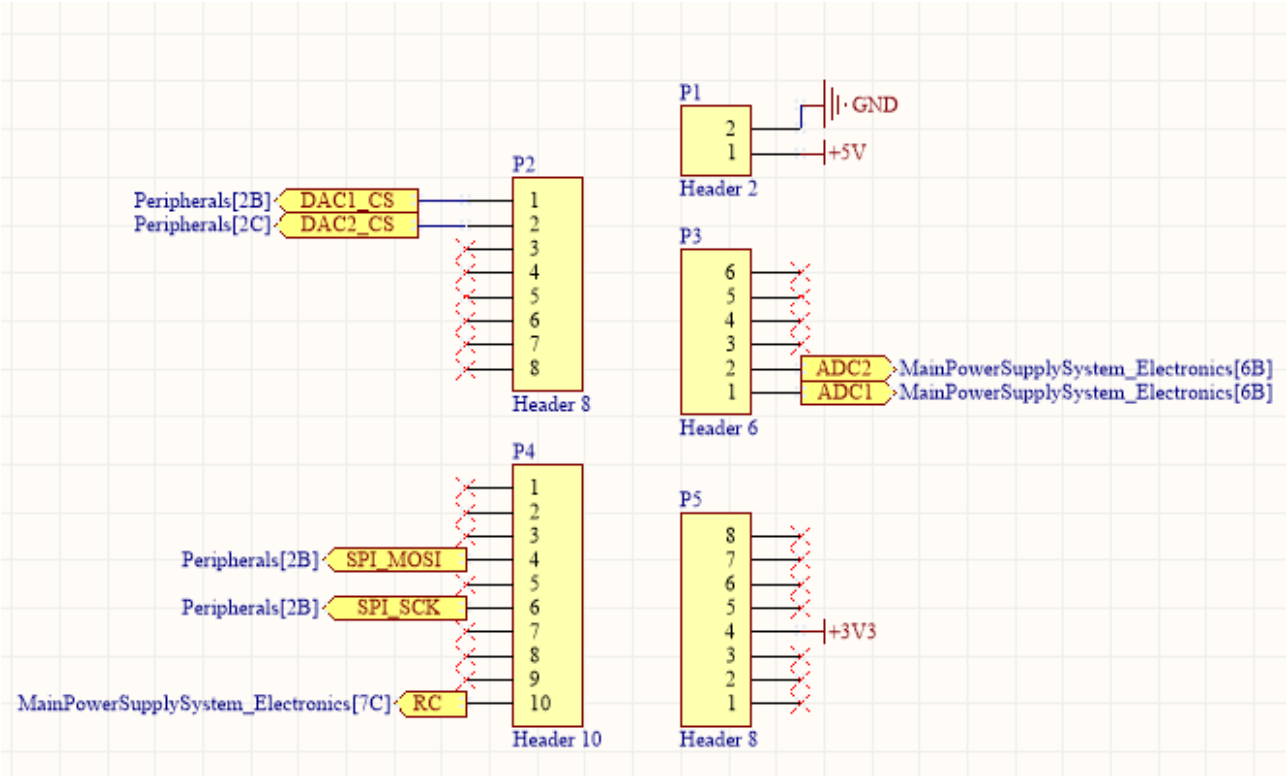


Figure 58: STM32F7 Pin connections

These schematics are then used to create a PCB layout. The final layout that was sent for printing is seen in Figure 59. The board size and shape was designed to perfectly fit the back of the STM32F746-Discovery board. The header layout was also done to match the Disco board exactly. This meant that the PCB would fit exactly on the back of the Disco board and thus reduce the size of the final design.

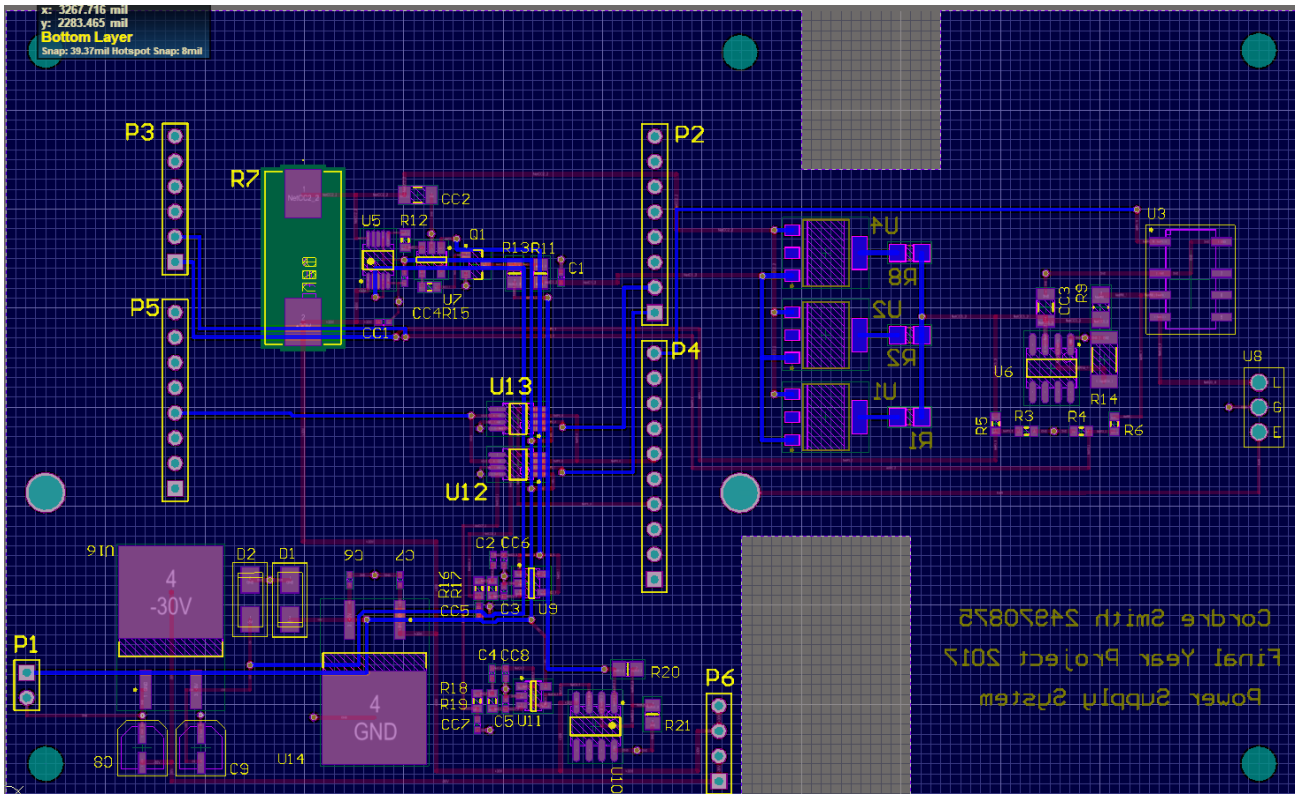


Figure 59: PCB layout

The PCB layout is then sent to China, where it was printed. The printed board was received about 4 weeks later. The PCB is then populated using the components that were also ordered earlier. The populated board is shown in **Error! Reference source not found.**.

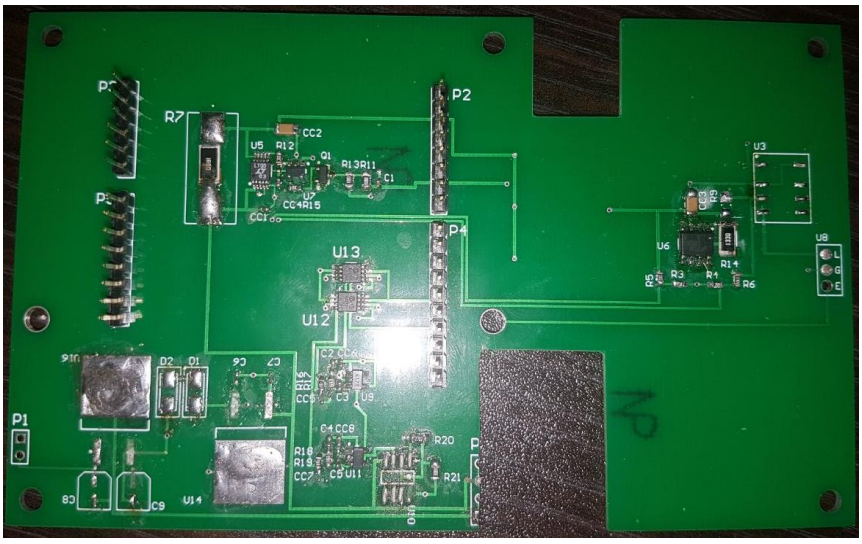


Figure 60: Populated PCB

4.6. Final Implementation

The PCB was finalized and tested, due to the negative regulator not working as expected and some manufacturing issues the PCB was badly damaged. The extent of the damage was impossible to determine. Thus the whole PCB had to be scrapped. This happened too close to final demonstration day, and no new PCB or components could be ordered. This meant that the proof of concept, prototype board (Veroboard) was used for the final implementation. The final implementation is seen in Figure 61.

Figure 61: Final implementation

Due to the Veroboard being implemented the overall size of the finale implementation increased slightly. The original casing also had to be scrapped, and thus a casing had to be manually built. Thus the final implementation did not look as great as it could. However, even with all of these problems the final product performed better than was expected. Prove of this statement together with the results of the testing of the final implementation is provided in the next Chapter.

5. Test and Evaluation

In the following Chapter the testing procedure, testing results and performance evaluation is discussed. The testing was done in steps. This Chapter is divided into sections with each section discussing one test procedure and its results.

5.1. Test the SMPS to check if correct output is received

The SMPS was tested by connecting it up correctly and plugging it into a wall socket. The output was then tested with the help of a multimeter. In Figure 62 the resulting voltage is seen. This shows that the SMPS outputs the correct DC voltage.

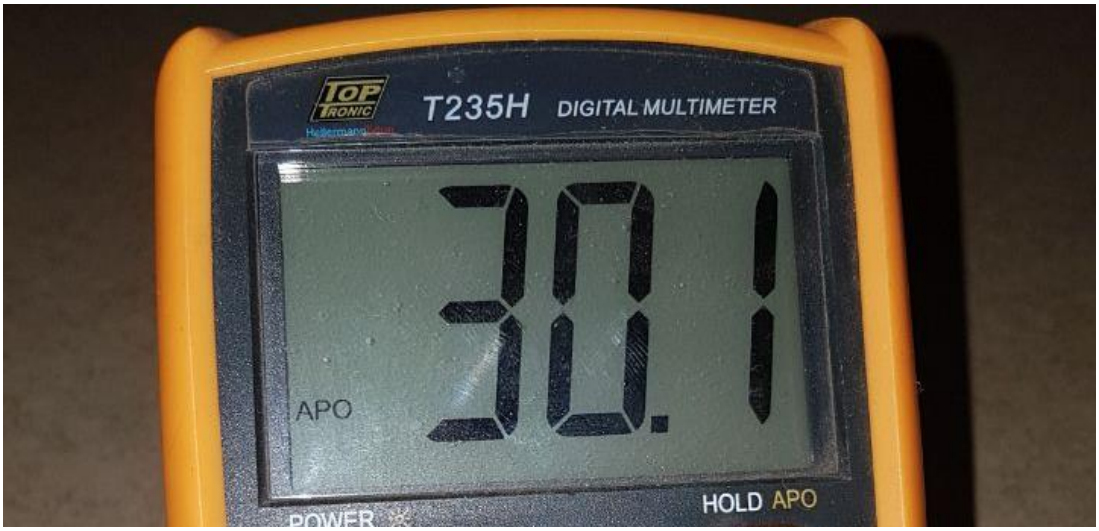


Figure 62: SMPS Output test

5.2. Test prototype board (Veroboard) with a Lab supply as power source

The Veroboard was connected up with a Lab supply as the power source, the set points were adjusted on the GUI, and the results were measured. Unfortunately, no photo evidence of this is available. However, in the digital archive DVD under Multimedia there is a video with the name *Veroboard_Test_Labsupply.mp4* this video serves as evidence of the working prototype.

5.3. Test prototype board (Veroboard) with SMPS as power source

The Veroboard was connected with the SMPS serving as the power source, the set points were again set using the GUI, and the results were measured! However this time current limiting of the PSS was also tested. This was unable to be tested in the previous section due to the Lab supply being implemented. Due to this setup being used as the final implementation the results of this test is shown in Section 5.5.

5.4. Test PCB

The PCB was populated, a lab supply was connected to power supply, set points were adjusted by the GUI, and the PCB was tested. However, with some testing of voltage present on the PCB using the multimeter it was determined that the Negative Regulator (L7905) that was used had some design or manufacturing issues and it was not producing negative five volts, but rather negative 11 volts. It was also later discovered that due to, small components being implemented there had been some manufacturing problem when soldering the PCB. These two combined issues made it impossible to determine which part of the PCB was damaged and the whole PCB had to be scrapped as said in the previous Chapter.

5.5. Test final implementation

The final implementation consisted of a Casing that contained all the components, with only the GUI, two banana connectors and main power source plug being visible. The final implementation is seen in Figure 61. The test setup used is seen in Figure 63.

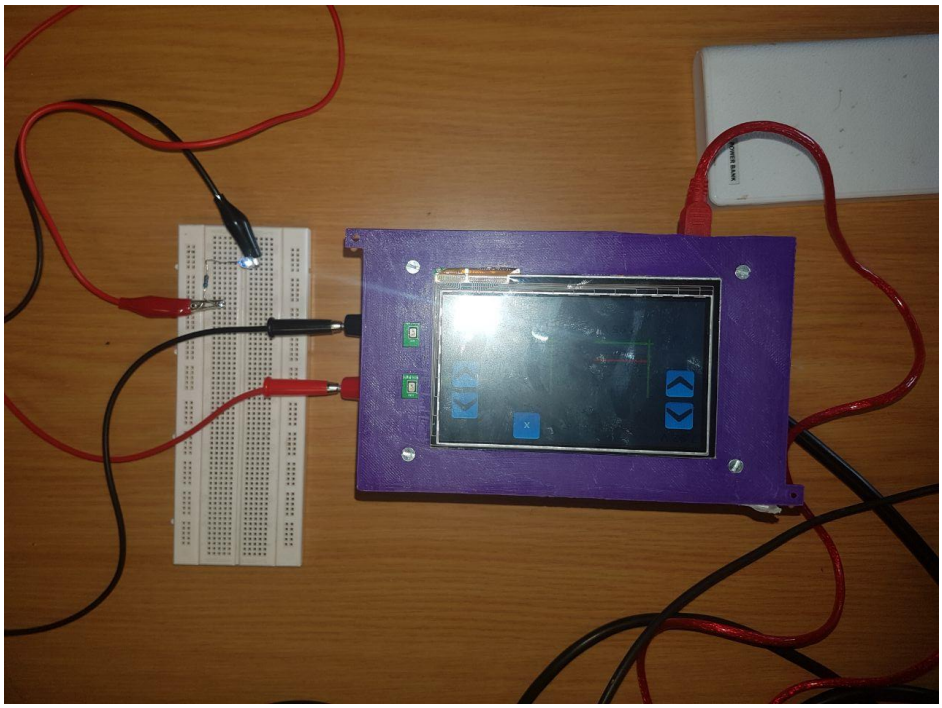


Figure 63: Final implementation testing setup

The final implementation was tested at 4 separated commonly used voltages (3 V, 5 V, 12 V and 24 V). The implementation was tested under conditions that should imply current limiting. Finally, a Fourier transform was also done to test the PSS with the SMPS as power supply vs a lab supply as the power supply to determine the amount of noise present on the PSS.

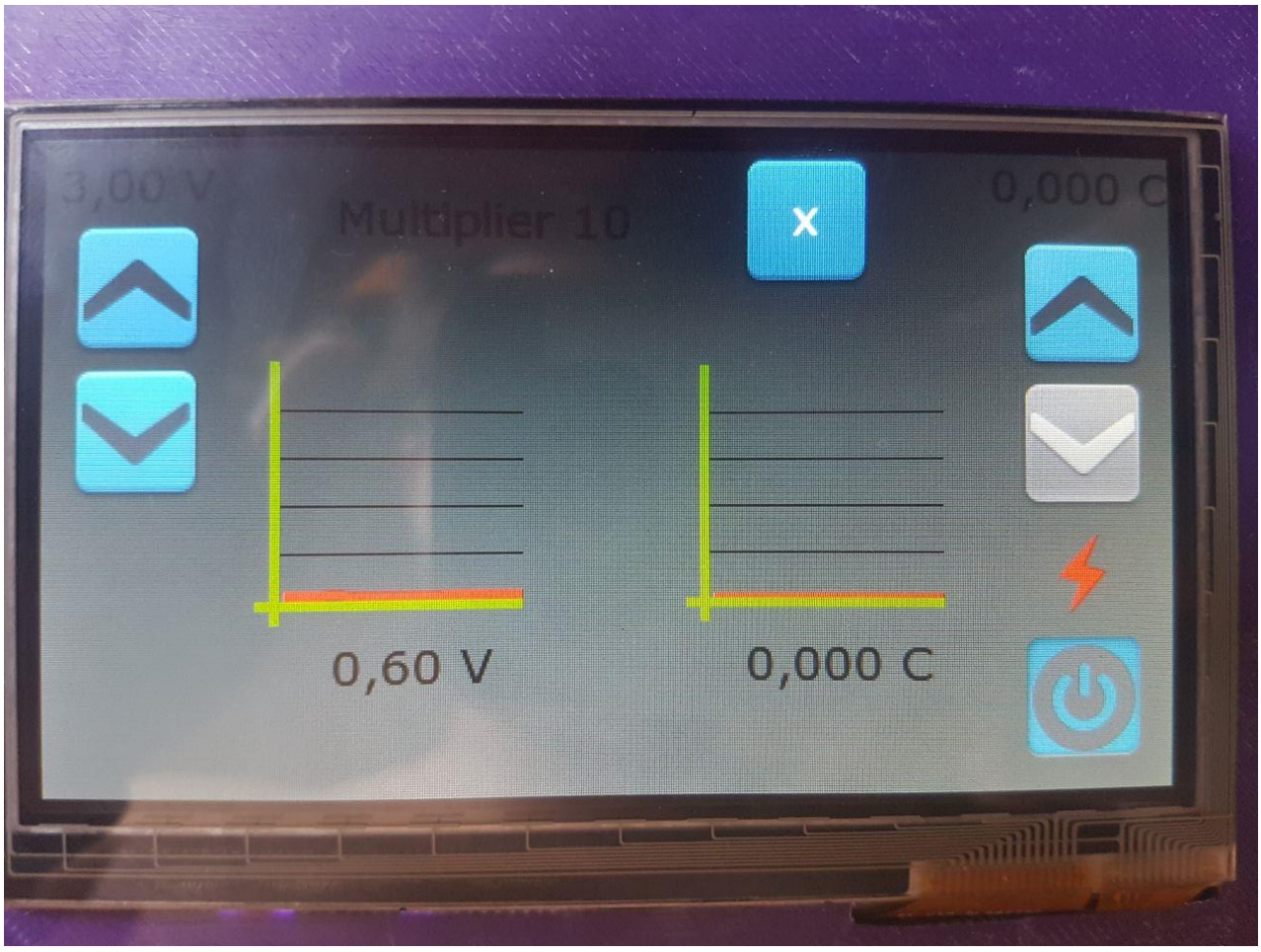


Figure 64: Current limiting test GUI

Firstly the current limiting feature of the PSS is tested. This is visually shown by the use of 3 different photos. Firstly in Figure 64 the GUI to show the setpoint values, secondly in Figure 65 the voltage meter reading over the load, and lastly in Figure 66 the output circuit that contains a LED that should not be shining, due to the current limit being set to low for the LED to function.

Figure 65: Current limiting test Multimeter

As is seen from Figure 65 the output voltage is not close to the set voltage shown in Figure 64; this is due to the PSS current limiting.

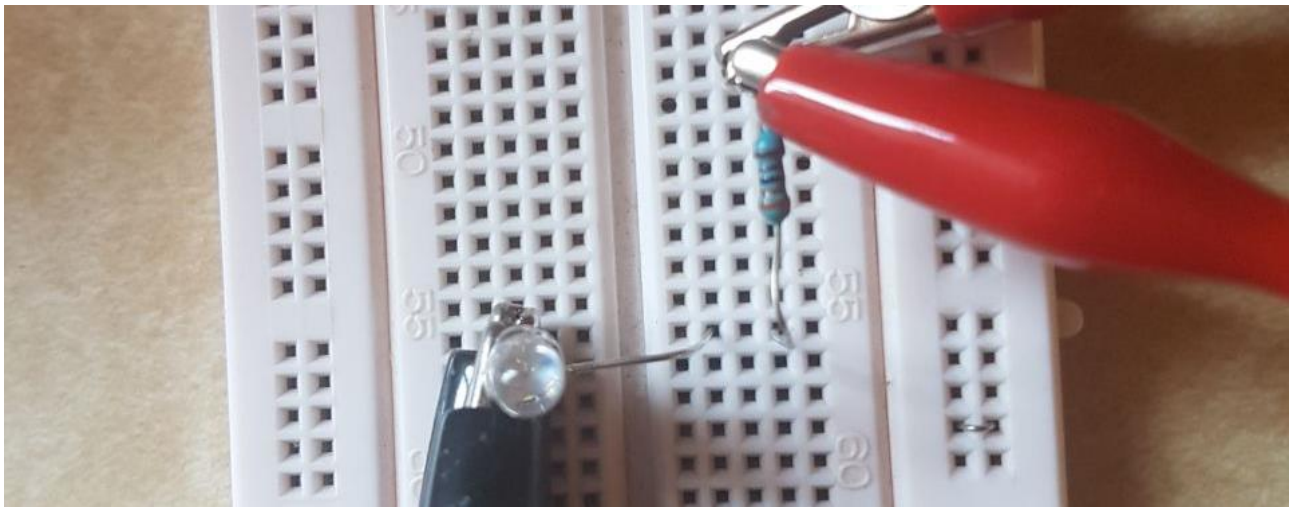


Figure 66: Current limiting test circuit

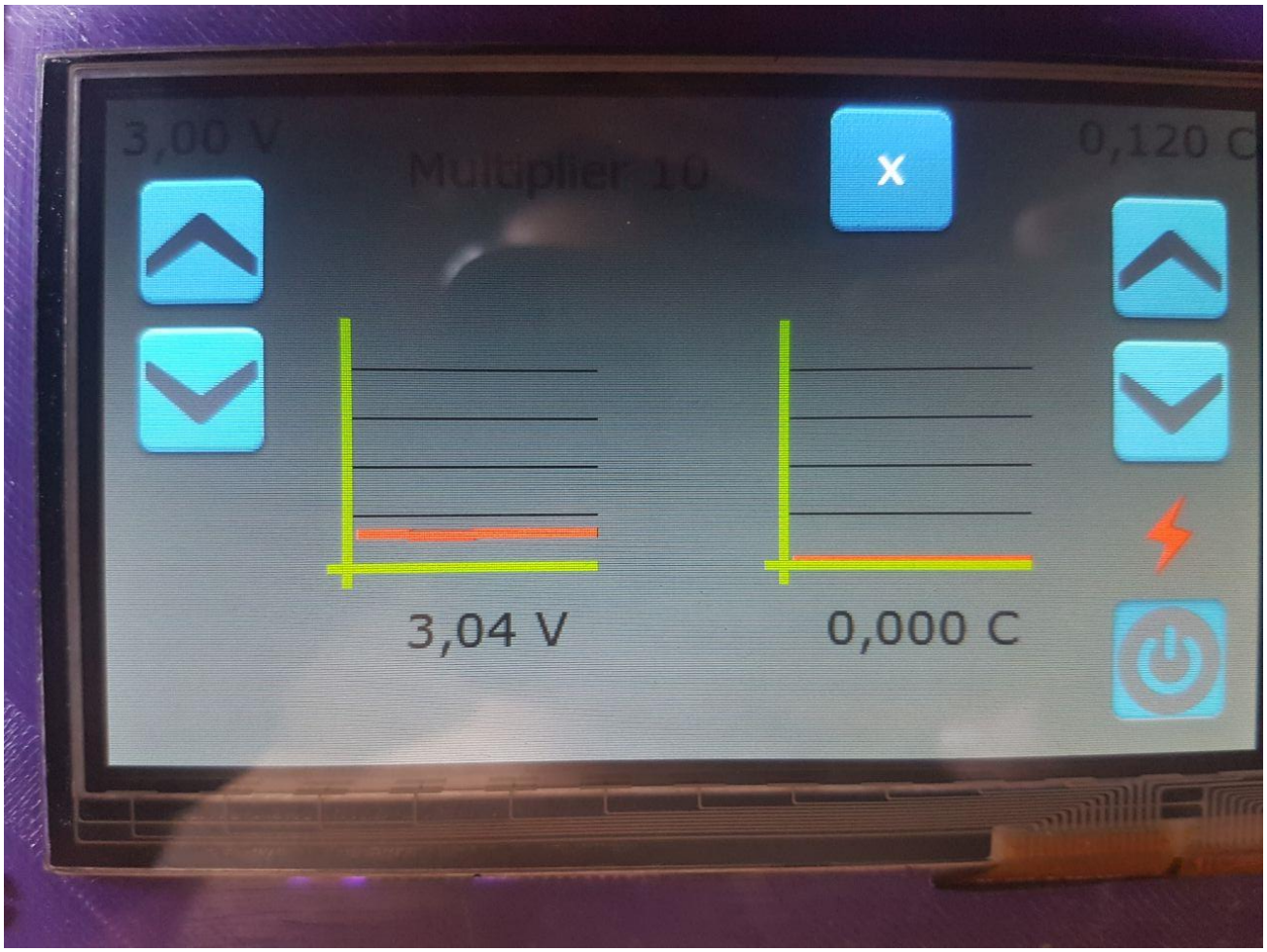


Figure 67: 3V test GUI

In Figure 66 the circuit for the current limiting is shown, and the result is as suspected. The LED is not emitting light due to the current being limited. This serves as to prove that the current limit in working order is. If video evidence is desired, the file *Current_Limiting_Test.mp4* under the Multimedia section on the Digital Archive will suffice.

In Figure 67 the set points for the 3V test is seen, the current limit has also be increased enough to allow for the LED to turn on.

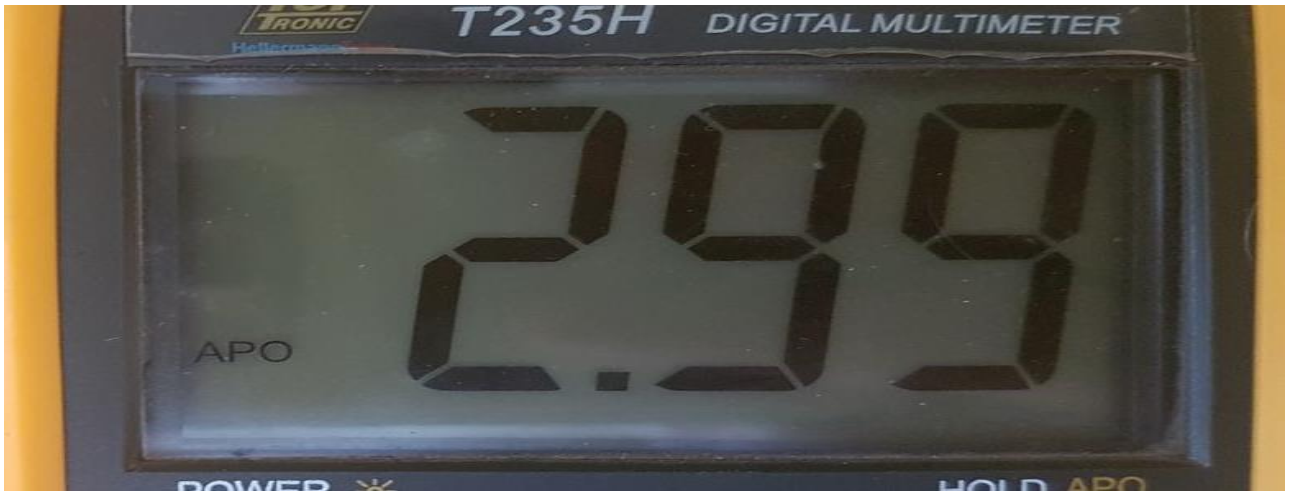


Figure 68: 3V test Multimeter

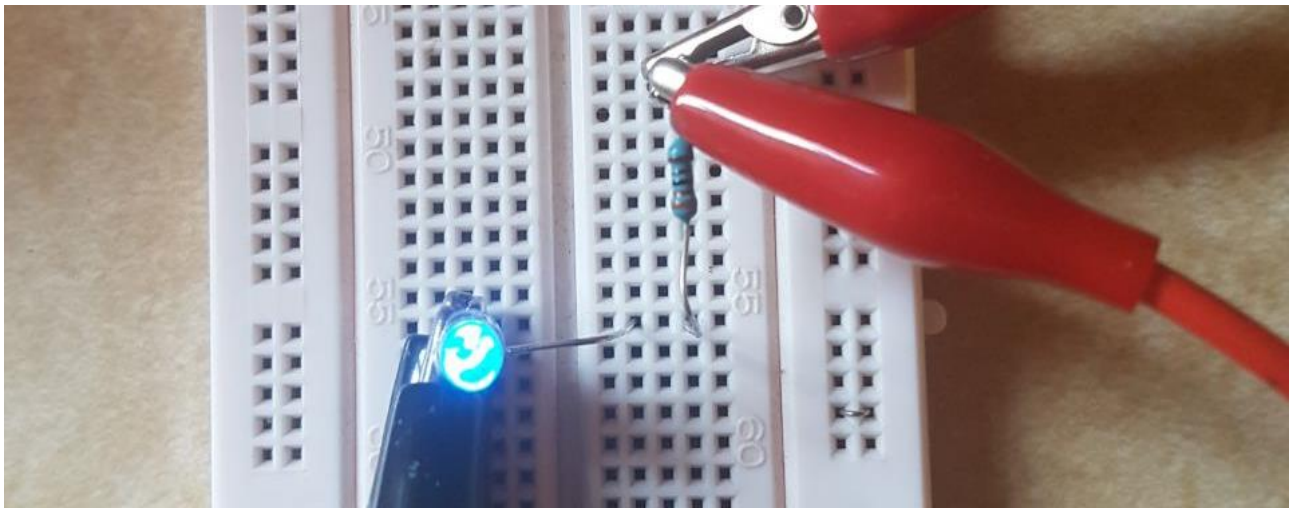


Figure 69: 3V test Circuit

In Figure 68 the multimeter reading of the output is seen, and in Figure 69 the output circuit with the suspected result is seen. For the next 3 testing voltage, the pictures have been excluded from the main report if these photos are sought after it is available in the Digital Archive under the Multimedia folder and named according to the following convention *VoltageLevel_Test_Type*.The results of the test have been tabulated in Table 6:

The final test that was executed was a Fourier Transform of the output when the Voltage Supply is the SMPS vs a lab supply. The Resulting graphs are seen in Figure 70 that was done for frequencies up to 1.25 kHz and then Figure 71 that was done for frequencies up to 500 kHz.

Table 6: Results

Testing Voltage (V)	Actual set Voltage (V)	Voltage reading
3	3	2.99
5	5.04	5.01
12	12	11.85
24	24	23.7

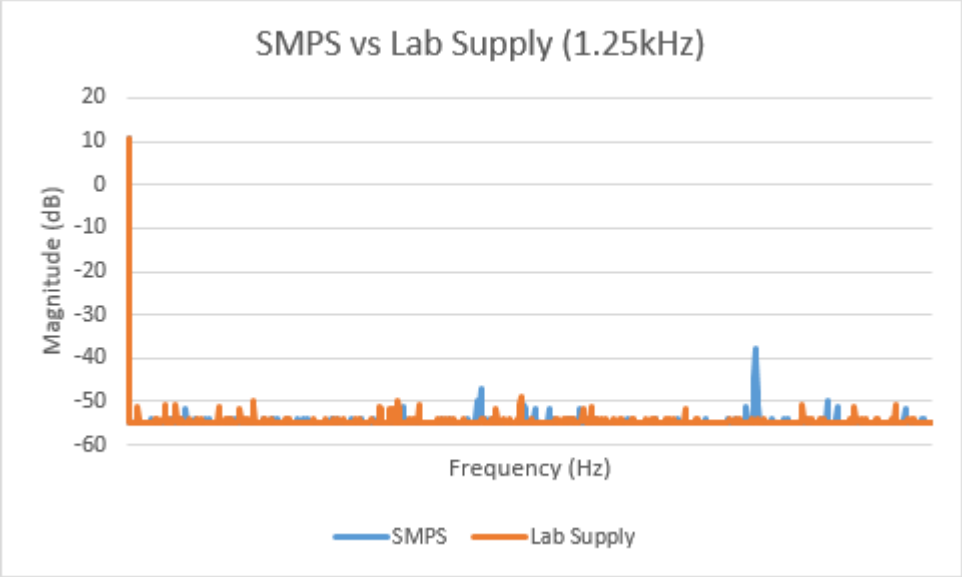


Figure 70: Fourier transform of output (1.25 kHz)

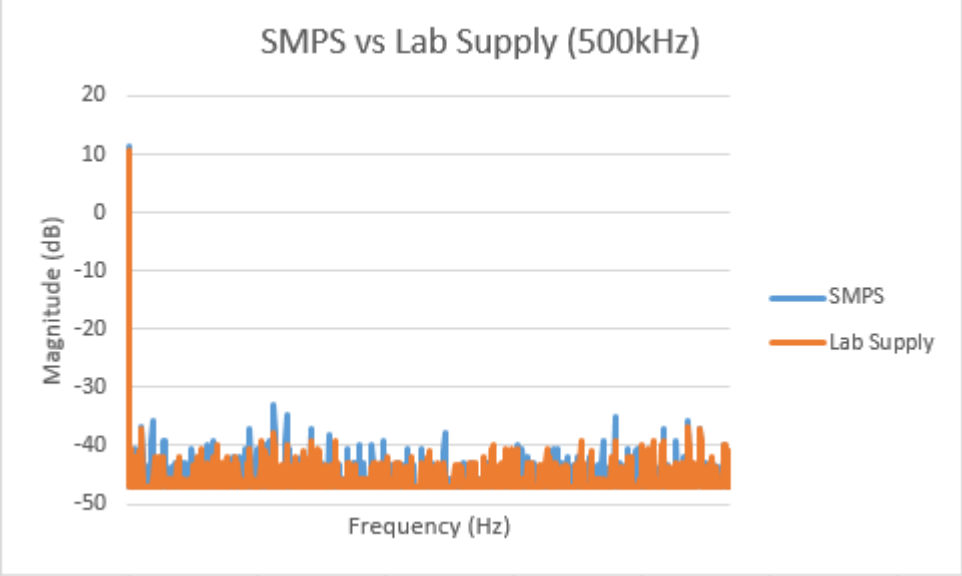


Figure 71: Fourier transform of output (500 kHz)

6. Conclusion and Recommendation

6.1. Conclusion

The project aimed to develop a Cost-Effective, Portable, a Digitally Controlled Benchtop Power supply that is also user-friendly. The power supply should exhibit basic power supply functions, namely Constant Voltage and Constant Current, and give students the opportunity to do off-campus development of practical's.

The currently available options are cheap Chinese versions. That is not digitally controlled; they are not very portable weighing 7 – 10 kg. They are also not very reliable due to their origin. The other option is to go for a heavy duty programmable power supply. They are digital and is reliable however one of these suppliers will cost you upwards of R7000 and is not close to being classified as portable.

The intended solution was to implement a Switch mode AC/DC power supply. This will reduce the size and weight significantly. However, it will produce noise in the PSS on the switching frequencies of the SMPS. The controlling circuitry is handled by a linear regulator to reduce the noise caused by the SMPS, and the interface will implement a TFT LCD integrated with a microcontroller to control the power supply digitally. The final touches will include implementing TouchGFX for the graphics on the LCS, implementing FreeRTOS to create a real-time system and increase the user-friendliness and finally the integration between the analogue and digital system is handled using ADC and DAC.

The final product was implemented on a Veroboard due to some complications with the soldering and components on the PCB. Due to the PCB being damaged the DACs was replaced by PWM signals. This reduced the fineness of the set points. The final implementation was added into a housing with banana connectors, GUI and the main power plug being the only intractable parts (thus limiting the hazards of shocking). The final implementation was tested, and the product exhibited current limiting, constant voltage and lower than expected noise on the output.

6.2. Recommendations

First of all the main recommendations has to be to correcting the PCB layout, reordering and populating a PCB properly and implementing DACs in place of the current PWM signals. These two alterations will increase the stability and reduce the noise and ripple of the system immensely.

Secondly, a proper housing should be designed and either 3D printed or laser cut to create a sturdier and more rigid housing that would increase the protection of the internal components and circuits greatly. This would improve the portability aspect of the system immensely.

The next few recommendations are extras that are done to improve the project. However, they are outside the scope of the original project and are thus just improvements.

These improvements include adding a customizable GUI to the system, where the user can specify new backgrounds, colour schemes and graphics. This will increase the likeability of the system by giving it a personal feel.

Next, the whole system is sold and set up as a DIY kit, with the main parts being provided. However, the user needs to do the soldering and assembling self. This will open up a market of new potential clients. Together with this, even though the project is intended for students, the project can also easily be marketed and sold to working-class engineering and gadgeteers.

The SMPS can also be designed self. This will take some effort and time testing and designing. However if done properly this can reduce the prize of the final product immensely, and the size of the product can also be reduced.

6.3. Adherence to ECSA Exit Level Outcomes

The ECSA ELO's that had to be showcased in this report and project are:

- ECSA ELO 1 – Solving engineering problem;

From the complexity of the problem that had to be solved and the accurate results that were achieved it is clear that ELO 1 has been done properly and correctly.

- ECSA ELO 3 – Engineering design and synthesis;

From Chapter 3 it is clear that design has been implemented and that the System Engineering Methodology is followed properly and thoroughly. From the circuit design and final design, the integration between different parts are clear and prominent this shows that synthesis was also done. The implementation of an RTOS and decisions made to achieve the desired result also shows that design had to be implemented.

- ECSA ELO 4 – Investigation, experiments and data analysis;

In Chapter 1 research into other options are done, the intended solutions were also simulated, and a prototype was built to show that it will work and finally the results were analysed and patterns and improvements were discovered. All these parts showed that ECSA ELO 4 had been completed properly.

- ECSA ELO 5 – Engineering methods, skills, tools and information technology;

From Altium, LTspice, Visual Studios, Keil MDK uVision5, MinGW compiler and STLink being implemented, it is rather clear that ECSA ELO 5 has been met.

- ECSA ELO 6 – Professional and technical communications;

From the above document and the presentation that was delivered on demo day, it is clear that Professional and Technical communications were done. All the other assignments throughout the year and presentations also show these skills.

- ECSA ELO 9 – Independent learning ability.

Learning how to implement a never used, on the NWU North West Campus, before graphical framework. Learning how to implement a Real Time Operating System. Learning how to use Altium to do PCB layouts. Learning proper (maybe not as good as what was needed) soldering skills. Working on reducing noise and ripple in a system. Designing an active filter. Integrating C and C++ code into one project.

All these are subjects that were learned and mastered to a degree of being able to finish the project. This shows and the ability for independent learning.

References

[1] CircuitSpecialists, '0-30V - 0-5A Linear Bench Power Supply'. [Online]. Available: https://www.circuitspecialists.com/csi3005t.html.

[2] Mouser, 'Benchtop Power Supplies Triple Output DC Power Supply 30V/3A'. [Online]. Available: https://www.mouser.co.za/ProductDetail/BK-Precision/9129B/?qs=sGAEpiMZZMvGBr7uFZP1K35ugFdjpImfospI4OwD6Ef1mq5s4NyVAQ%3D%3D.

[3] I. Sommerville, 'Software engineering 6th Edition', 2000.

[4] 'Electricity in South Africa: about volts, amps and plugs – Travel Tips'. [Online]. Available: http://www.southafrica.net/za/en/travel-tips/entry/travel-tip-electricity. [Accessed: 28-Feb-2017].

[5] 'Potchefstroom, South Africa Travel Weather Averages (Weatherbase)'. [Online]. Available: http://www.weatherbase.com/weather/weather.php3?s=5386&cityname=Potchefstroom-North-West-South-Africa&units=metric. [Accessed: 28-Feb-2017].

[6] E. Brorein, 'Should I use a Switching or Linear DC Power Supply for my next test system? (part 1 of 4)', *Keysight Technologies' Power Blog*, 2011. [Online]. Available: http://powersupply.blogs.keysight.com/2011/11/should-i-use-switching-or-linear-dc.html. [Accessed: 28-Feb-2017].

[7] R. Ahmed and M. Jahangir, 'Power Quality Improvement Using Switch Mode Regulator', in *An Update on Power Quality*, InTech, 2013.

[8] E. Brorein, 'Should I Use a Switching or Linear DC Power Supply For My Next Test System? (part 2 of 4)', *Keysight Technologies' Power Blog*, 2011. [Online]. Available: http://powersupply.blogs.keysight.com/2011/11/should-i-use-switching-or-linear-dc_23.html. [Accessed: 28-Feb-2017].

[9] 'Transformer Basics and Transformer Principles'. [Online]. Available: http://www.electronics-tutorials.ws/transformer/transformer-basics.html. [Accessed: 28-Feb-2017].

[10] 'Full Wave Rectifier and Bridge Rectifier Theory'. [Online]. Available: http://www.electronics-tutorials.ws/diode/diode_6.html. [Accessed: 28-Feb-2017].

[11] 'Series Voltage Regulator | Series Pass Regulator | Tutorial'. [Online]. Available: http://www.radio-electronics.com/info/power-management/linear-power-supply-psu/series-voltage-regulator-theory-circuit.php. [Accessed: 28-Feb-2017].

[12] H. J. Zhang, 'Basic Concepts of Linear Regulator and Switching Mode Power Supplies', *Linear Technol. Appl. Note*, vol. 140, no. October, pp. 1–16, 2013.

[13] ‘What is Digital Signal Processing (DSP)? - Definition from Techopedia’. [Online]. Available: https://www.techopedia.com/definition/2360/digital-signal-processing-dsp. [Accessed: 28-Feb-2017].

[14] ‘DAC3151 10-bit, 500-MSPS Digital-to-Analog Converter (DAC) | TI.com’. [Online]. Available: http://www.ti.com/product/dac3151/datasheet. [Accessed: 28-Feb-2017].

[15] Texas Instruments, ‘ADC1175 8-Bit, 20MHz, 60mW A/D Converter Check for Samples: ADC1175’, 2000.

[16] ‘STM32 MCU Discovery Kits - STMicroelectronics’. [Online]. Available: http://www.st.com/en/evaluation-tools/stm32-MCU-discovery-kits.html?querycriteria=productId=LN1848$$4294=STM32F4. [Accessed: 28-Feb-2017].

[17] ‘STM32F405xx, STM32F407xx’.

[18] ‘STM32 32-bit ARM Cortex MCUs - STMicroelectronics’. [Online]. Available: http://www.st.com/en/microcontrollers/stm32-32-bit-arm-cortex-mcus.html?querycriteria=productId=SC1169. [Accessed: 28-Feb-2017].

[19] ‘What is LCD (liquid crystal display)? - Definition from WhatIs.com’. [Online]. Available: http://whatis.techtarget.com/definition/LCD-liquid-crystal-display. [Accessed: 28-Feb-2017].

[20] ST Microelectronics, ‘Description of STM32F7 HAL and Low-layer drivers’, no. February 2017.

[21] STMicroelectronics, ‘STM32CubeMX User Manual’, vol. 0, no. October, pp. 1–280, 2017.

[22] Draupner Graphics A/S, ‘TouchGFX’, 2017. [Online]. Available: www.touchgfx.com.

[23] Microsoft, ‘Visual Studios Community 2015’. 2015.

[24] ARM, ‘Keil MDK-Lite uVision 5’. 2016.

[25] Altium Limited, ‘Altium Designer’. 2017.

[26] Linear Technology, ‘LT3080 - Adjustable 1.1A Single Resistor Low Dropout Regulator’, vol. LT 0911 RE, pp. 1–28.

[27] Linear Technology, ‘LT1991 - Precision, 100µA Gain Selectable Amplifier’, pp. 1–28, 2006.

[28] Texas Instruments, ‘LM397 Single General-Purpose Voltage Comparator’, 2016.

[29] ON Semiconductor, ‘SMMBT2222AL General Purpose Transistors’, pp. 1–7, 2016.

[30] Texas Instruments, ‘LM134 / LM234 / LM334 3-Terminal Adjustable Current Sources’, no. March 2000, 2013.

[31] Bel Power Solutions & Protection, ‘ABC120 Series’.

[32] C. Data, ‘IM Relay 2 Pole Cha IM Relay 2 Pole Change’.

[33] STMicroelectronics, ‘Positive voltage regulators - L78’, no. March, pp. 1–58, 2014.

[34] STMicroelectronics, ‘Negative voltage regulators - L79’, no. September, pp. 1–13, 1999.

[35] M. Engelhardt, ‘LTspice’. Linear Technology Corporation, 2017.

[36] ON Semiconductor, ‘LM258, LM358, LM358A, LM358E, LM2904, LM2904A, LM2904E, LM2904V, NCV2904’, *Order A J. Theory Ordered Sets Its Appl.*, pp. 1–14, 2003.

[37] STMicroelectronics, ‘STM32F746-Disco User Manual’, no. June 2017.

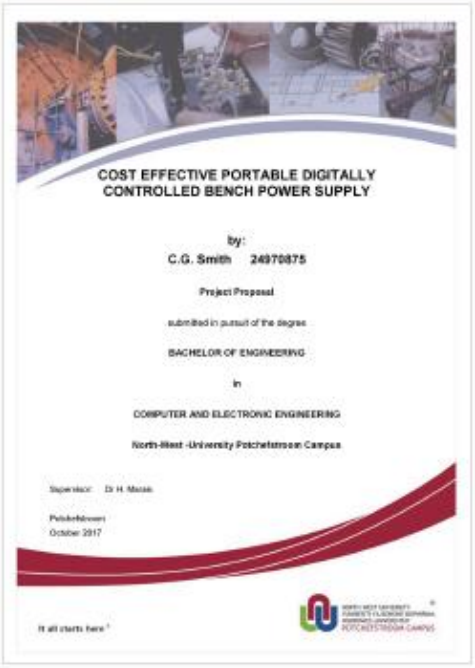
turnitin

Digital Receipt

This receipt acknowledges that Turnitin received your paper. Below you will find the receipt information regarding your submission.

The first page of your submissions is displayed below.

Submission author:	CORDRÈ SMITH
Assignment title:	Final Report for Plagiarsm Check -...
Submission title:	CG_Smith_24970875_Final_Repor...
File name:	CG_Smith_24970875_Final_Repor...
File size:	3.57M
Page count:	79
Word count:	16,872
Character count:	97,884
Submission date:	30-Oct-2017 02:01PM (UTC+0200)
Submission ID:	871283114



COST EFFECTIVE PORTABLE DIGITALLY CONTROLLED BENCH POWER SUPPLY

by:
C.G. Smith 24970875

Project Proposal

submitted in pursuit of the degree

BACHELOR OF ENGINEERING

in

COMPUTER AND ELECTRONIC ENGINEERING

North-West -University Potchefstroom Campus