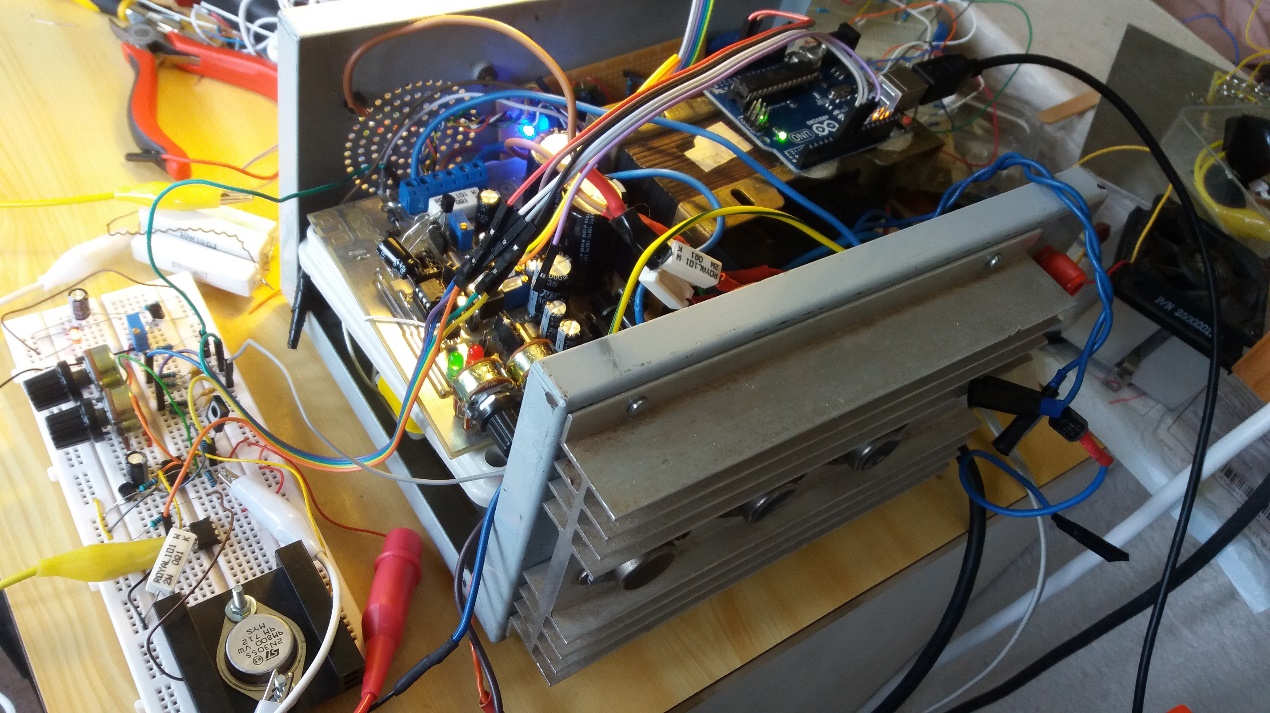
10/1/2016

Daniel Robinson

Stellenbosch University

Linear Power-supply Design

Electronic-Design 344



# Declaration

I, the undersigned, hereby declare that the work contained in this report is my own original work and unless otherwise stated.

Signature: ...................................

D. Robinson

Date: ...........................................

# Abstract

This report will document the design, analysis, measurement and correlation between theoretical, simulated and measured values of a power supply which can be digitally interfaced, per desired specifications. Due to the limited time frame, and the amateur experience, it is indeed a simple power supply; also, meant to educate the designer.

The report will document all the obstacles, errors and choices made, until a relatively stable conclusion is reached, throughout the paper.

# Summary

This project was built using a huge 160W transformer in mind, but due to the large currents it damaged the PCB beyond repair within the remaining time left for the project. In fact, it was the morning of the demo that it happened. A quick simple circuit was breadboarded and demonstrated later that day. Both designs will be discussed throughout the paper.

One highlight was sourcing 19A through that 160W transformer

Table of Contents

[Declaration 1-1](#_Toc465035011)

[Abstract 1-2](#_Toc465035012)

[Summary 1-3](#_Toc465035013)

[1 List of Abbreviations 1-8](#_Toc465035014)

[2 Introduction 2-9](#_Toc465035015)

[3 Problem Statement 3-10](#_Toc465035016)

[4 Literature Study: System 4-11](#_Toc465035017)

[5 System Design 5-11](#_Toc465035018)

[5.1 Transformers 5-11](#_Toc465035019)

[5.1.1 Literature Study 5-11](#_Toc465035020)

[5.1.2 Design 5-11](#_Toc465035021)

[5.1.3 Analysis 5-12](#_Toc465035022)

[5.1.4 Building the circuit 5-12](#_Toc465035023)

[5.1.5 Measurements 5-12](#_Toc465035024)

[5.1.6 Comparison of theoretical and measured values 5-12](#_Toc465035025)

[5.1.7 Conclusion and recommendations 5-12](#_Toc465035026)

[5.2 Rectifiers & Capacitor Banks 5-13](#_Toc465035027)

[5.2.1 Literature Study 5-13](#_Toc465035028)

[5.2.2 Design 5-13](#_Toc465035029)

[5.2.3 Analysis 5-14](#_Toc465035030)

[5.2.4 Building the circuit 5-14](#_Toc465035031)

[5.2.5 Measurements 5-14](#_Toc465035032)

[5.2.6 Comparison of theoretical and measured values 5-14](#_Toc465035033)

[5.2.7 Conclusion and recommendation 5-14](#_Toc465035034)

[5.3 Zener Constant-Voltage Reference 5-14](#_Toc465035035)

[5.3.1 Literature Study 5-14](#_Toc465035036)

[5.3.2 Design 5-14](#_Toc465035037)

[5.3.3 Analysis 5-15](#_Toc465035038)

[5.3.4 Building the circuit 5-15](#_Toc465035039)

[5.3.5 Measurements 5-15](#_Toc465035040)

[5.3.6 Comparison of theoretical and measured values 5-15](#_Toc465035041)

[5.3.7 Conclusion and recommendation 5-15](#_Toc465035042)

[Literature Study 5-15](#_Toc465035043)

[5.4 Outboard Pass transistor stage 5-15](#_Toc465035044)

[5.4.1 Literature Study 5-15](#_Toc465035045)

[5.4.2 Design 5-16](#_Toc465035046)

[5.4.3 Analysis 5-18](#_Toc465035047)

[5.4.4 Building the circuit 5-18](#_Toc465035048)

[5.4.5 Measurements 5-18](#_Toc465035049)

[5.4.6 Comparison of theoretical and measured values 5-18](#_Toc465035050)

[5.4.7 Conclusion and recommendation 5-18](#_Toc465035051)

[5.5 Voltage Regulator 5-19](#_Toc465035052)

[5.5.1 Literature Study 5-19](#_Toc465035053)

[5.5.2 Design 5-19](#_Toc465035054)

[5.5.3 Analysis 5-19](#_Toc465035055)

[5.5.4 Building the circuit 5-19](#_Toc465035056)

[5.5.5 Measurements 5-19](#_Toc465035057)

[5.5.6 Comparison of theoretical and measured values 5-19](#_Toc465035058)

[5.5.7 Conclusion and recommendation 5-19](#_Toc465035059)

[5.6 Current limiter 5-20](#_Toc465035060)

[5.6.1 Literature Study 5-20](#_Toc465035061)

[5.6.2 Design 5-20](#_Toc465035062)

[5.6.3 Analysis 5-20](#_Toc465035063)

[5.6.4 Building the circuit 5-20](#_Toc465035064)

[5.6.5 Measurements 5-20](#_Toc465035065)

[5.6.6 Comparison of theoretical and measured values 5-20](#_Toc465035066)

[5.6.7 Conclusion and recommendation 5-20](#_Toc465035067)

[6 Circuit Integration (Analogue) 6-21](#_Toc465035068)

[6.1 Final System Measurements 6-21](#_Toc465035069)

[6.2 Interpretation of Results 6-21](#_Toc465035070)

[7 Software Design 7-22](#_Toc465035071)

[7.1 Purpose & Requirement 7-22](#_Toc465035072)

[7.2 Software Literature 7-22](#_Toc465035073)

[7.3 Software Extras 7-23](#_Toc465035074)

[7.3.1 Control System 7-23](#_Toc465035075)

[7.3.2 Battery charging profiles 7-24](#_Toc465035076)

[7.3.3 Cellular connectivity 7-25](#_Toc465035077)

[7.4 Arduino Interface 7-25](#_Toc465035078)

[7.4.1 Inputs (PWM RC filters) 7-25](#_Toc465035079)

[7.4.2 Outputs 7-27](#_Toc465035080)

[8 Appendix A: Measured Demonstration Results 8-29](#_Toc465035081)

[9 Appendix B: Circuit Diagram 9-30](#_Toc465035082)

[10 Appendix C: PCB Layout 10-31](#_Toc465035083)

[11 Appendix D: Photo of Circuit 11-32](#_Toc465035084)

[12 Appendix E: Calculations 12-33](#_Toc465035085)

[12.1 Appendix E.1: Transformers 12-33](#_Toc465035086)

[12.2 Appendix E.2: Rectifiers & Capacitor Banks 12-33](#_Toc465035087)

[12.3 Appendix E.3: Zener Constant Voltage Reference 12-33](#_Toc465035088)

[12.4 Appendix E.4: Pass Output Stage 12-33](#_Toc465035089)

[12.5 Appendix E.5: Voltage Regulator 12-33](#_Toc465035090)

[12.6 Appendix E.6: Current Limiter 12-33](#_Toc465035091)

[12.7 Appendix E.7: Arduino Interface 12-33](#_Toc465035092)

[12.7.1 PWM filter 12-33](#_Toc465035093)

[13 Appendix F: Source code 13-35](#_Toc465035094)

[14 Appendix G: Extra information 14-47](#_Toc465035095)

[14.1 Voltage Gain 14-47](#_Toc465035096)

[14.2 Current Gain (transconductance) 14-47](#_Toc465035097)

[14.3 Extra Hardware Features 14-47](#_Toc465035098)

[14.3.1 Battery charging profiles 14-47](#_Toc465035099)

[15 Figures 15-47](#_Toc465035100)

[16 Glossary 16-49](#_Toc465035101)

[17 Bibliography 17-50](#_Toc465035102)

# List of Abbreviations

* PCB – Printed Circuit Board
* SMPS – Switched Mode Power Supplies
* NiMH – Nickel-Metal Hydride
* Pb – Lead
* Li – Lithium
* RC – Resistor Capacitor
* LPF – Low Pass Filters

# Introduction

The aim of this project is to build a power supply to satisfy a client’s requirements. It must have minimal ripple, as well as constant voltage and current mode. It must also be built using analogue components such as op amps, but not ICs such as voltage regulators (e.g. LM317). There are components provided, but it is up to the designer what they use. They may use components provided by the Stellenbosch University store room, or source their own.

The power supply is the heart of all electronic devices. An understanding is crucial in understanding the non-linear effects, responses and ways a power supply can affect a product. Power supply design is very broad, and two directions which benefit relatively equally, albeit differently, from a project like this, are either ultra-low powered, or high powered applications.

Lastly, there will be a limited time frame to complete this project, which simulates industry conditions, and is mostly meant to educate the designer him/herself, by overcoming pitfalls the designer is mostly unprepared for. Such pitfalls one really only overcomes through experience, and it is a good precursor to a future in the industrial or academic world.

[insert flow diagram here]

# Problem Statement

The design of the power supply will overcome the following problems:

* Low ripple
* It is required to provide 1A at 12V
* It is required to provide 500mA at 14V
* It is required to measure and set voltages/currents via a PC.
* (Optional) Charge NiH and Pb batteries using software profiles
* The op amps can output max Vcc – 1.5V
* The output stage pass transistors have a voltage drop across them (perhaps 1V).
* All in all, there is about a 2-3V drop to regulate unregulated voltage after the diode bridge.
* Non-linear effects of op amps, transistors, Zeners etc. must be considered

# Literature Study: System

Power supplies are used all over the world. There are many different kinds. Besides varying power ratings, they are divided into mainly two fields: linear and switched mode power supplies, with the latter being more complicated. A note on SMPS: due to its frequency selective behaviour in choosing output current for an arbitrary load, it is quite efficient, and requires much smaller transformers as opposed to linear power supplies.

On the other hand, SMPS are very noisy in radio applications, and rather require linear power supplies.

[Insert table of comparisons]

# System Design

This will include the design, analysis, measurements and comparisons of theory and results as well as an introductory literature study.

## Transformers

### Literature Study

*“Michael Bay invented transformers in 1903. “*

### Design

There are two transformers provided: a 230V primary to 15V 1A secondary transformer, as well as a dual secondary 9V 400mA transformer. There are another two transformers that the designer had available: a 16V dual secondary (unknown current rating), and a 20V 8A secondary transformer.



Figure 5‑1: 15V transformer secured

The 15V is supposed to provide the power, whilst the 9V is supposed to provide a stable differential input to the op amps. The reason why it might be suggested that the op amps have a differential supply is that it is specified in [insert TL081 datasheet reference] op amps made available to the Design 344 students that the output voltage is 1.5V from the rails. If one uses ground for the negative V- supply of the op amp, then the minimum output voltage is 1.5V.



Figure 5‑2: 9V dual transformer and two diode bridges (right)

The 16V dual secondary seemed like a nice bet, and due to the size, it was estimated that the current rating is about 2 to 3 Amps.

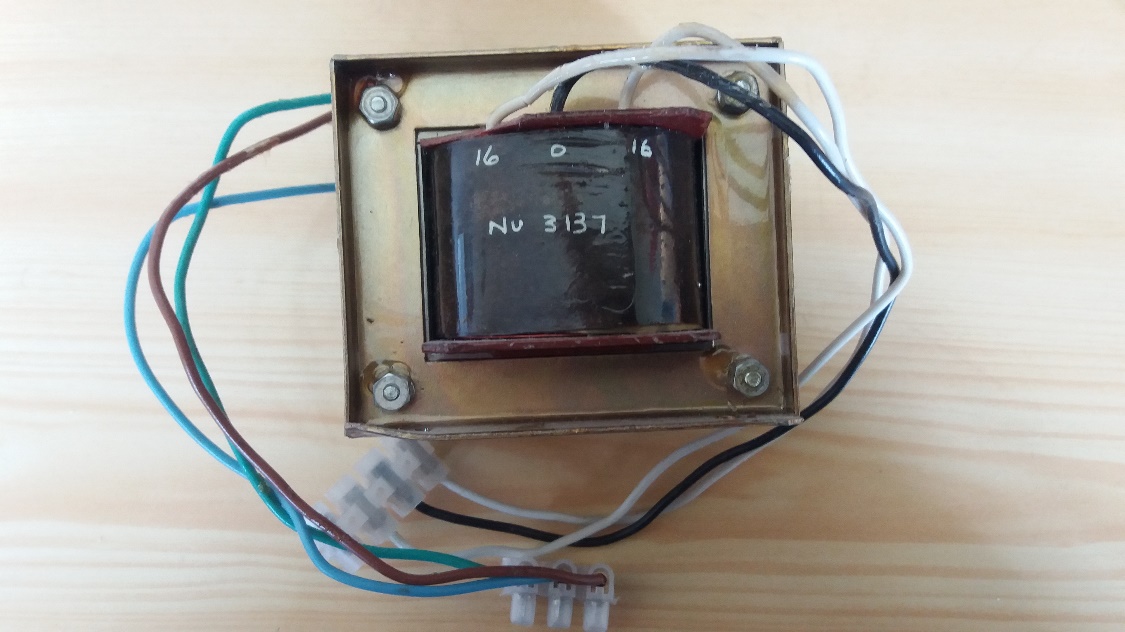


Figure 5‑3: The dual 16V transformer

However, the greatest one of them all is the 20V 8A one. In future, it will be referred to as the 160W transformer.



Figure 5‑4: 160W transformer

It was planned that dual LM358 op amps would be used since a bountiful supply was available, and they could output 0V in single rail mode [[1]](#footnote-1) when sinking low currents. The goal is just to design a power supply that requires only 1 rail to fully make use of the extra voltage overhead. Then one also only needs one transformer in the end.

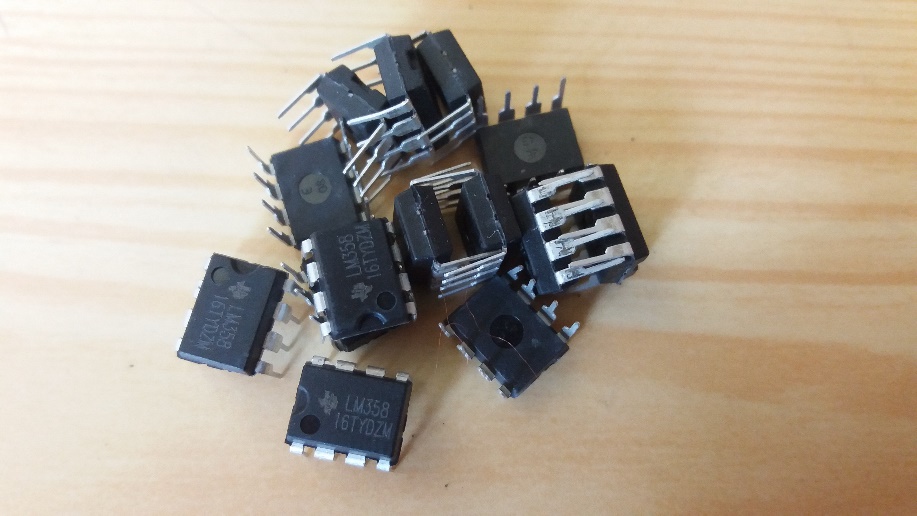


Figure 5‑5: Couple of dual LM358s

### Analysis

When the 15V transformer voltages were measured under load, it was discovered that Rseries was about 2.86 ohms. *See Appendix E.1*

Using this information, a spice model was built to see what kind of voltages to expect under 1 Amp load.

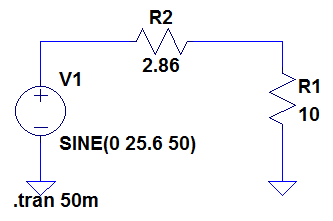


Figure 5‑6: simple thevinin equivalent 15V transformer model.

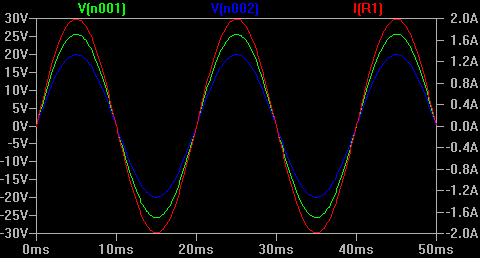


Figure 5‑7: voltage drop after series resistance (blue), current (red)

With an average current of 1A, we can see here that the peak voltage only drops about 5V. Unfortunately this is not very helpful without the capacitive load as well, since one does not yet know by how much the voltage will drop in the end.

### Building the circuit

Insert pictures of crimp connectors.

### Measurements

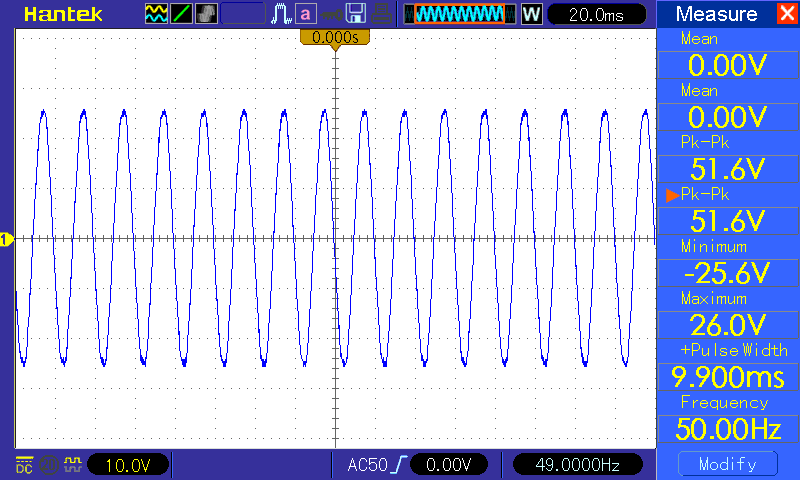


Figure 5‑8: 15V transformer open circuited

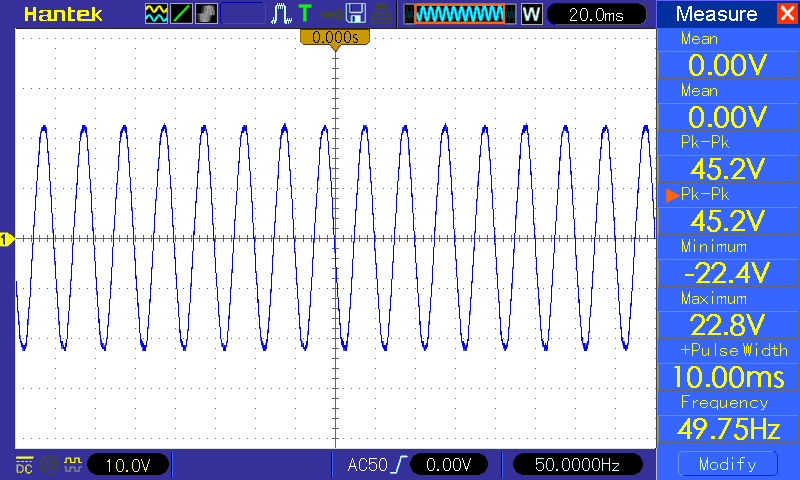


Figure 5‑9: 15V transformer loaded with 20 ohms

*See Appendix E.1 for calculations.*

Rseries = 2.86 ohms

### Comparison of theoretical and measured values

### Conclusion and recommendations

## Rectifiers & Capacitor Banks

### Literature Study

*“Today, tomorrow, together we’ll open a capacitor bank.”*

### Design

The Taylor series approximated ripple equation gives the voltage ripple over the capacitors when the max current is flowing through the diode bridge and depends on the frequency of the mains, and the capacitance used.

Two 10mF 35V capacitors were available. With Imax = 1A, f = 50Hz and C = 20mF.

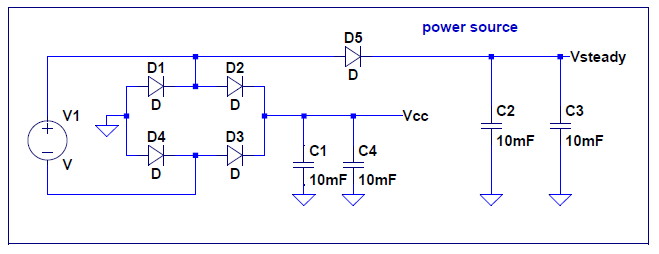


Figure 5‑10: Bridge and capacitors

A steadier voltage reference was used to supply power to everything except the pass transistor. With the estimated power usage to be about 100mA, the steadier voltage supply would have about 50mV of ripple. Besides current not being able to escape through the blocking diode, it is steadier because less current discharges out of the capacitors, and relates to the equation:

With less current, and more capacitance, the change in voltage is less.

Luckily, the whole purpose of a voltage regulator is to regulate the input. Therefore, the op amp can compensate for this lesser ripple by having a stable Zener reference.

### Analysis

Here is a circuit to display the two different voltages that will be used to power the circuit.

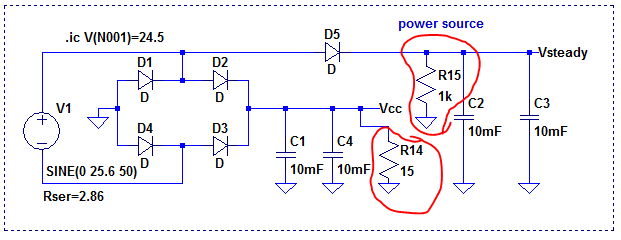


Figure 5‑11: 15 ohm load, and 1k represents the load of the rest of circuit

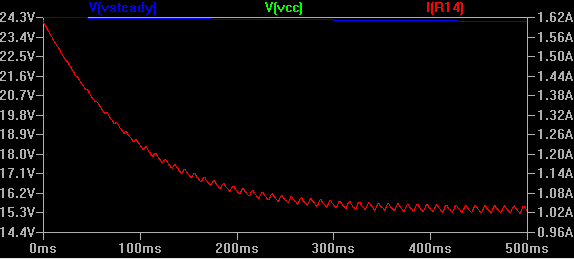


Figure 5‑12: Diode + capacitor bank under load.

It appears to be a neat trick. The voltage tapped off before the diode bridge seems to be much higher than the voltage used to power the load.

### Building the circuit

The circuit was built with a diode bridge which accepts crimp connectors. This means that hotswapping a transformer is very quick, easy, and requires no soldering.



Figure 5‑13: Old diode bridge

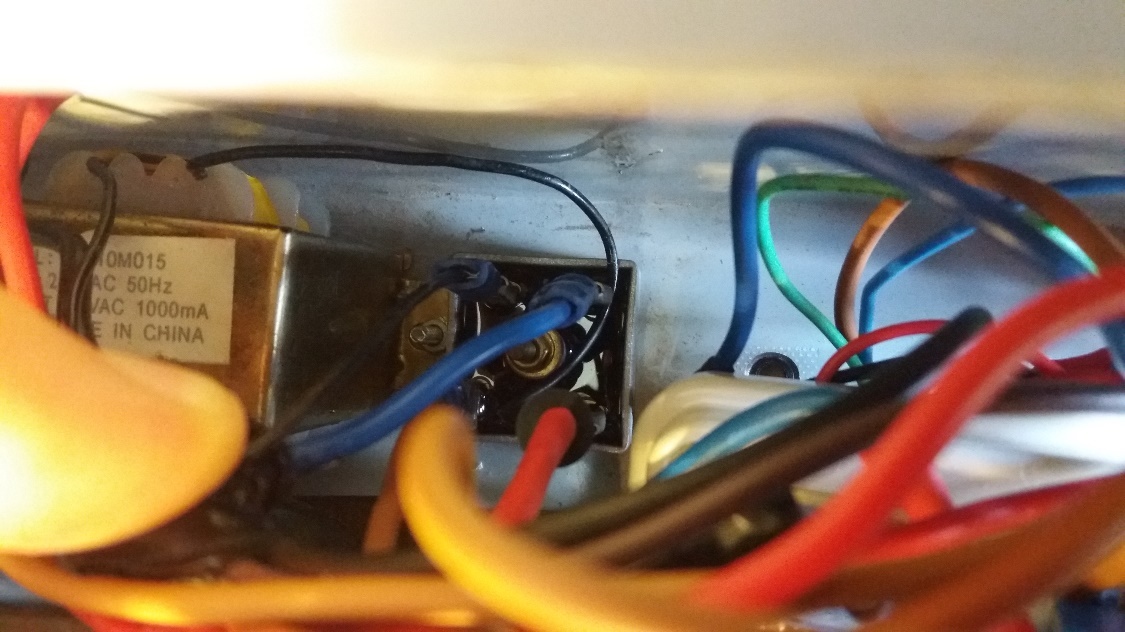


Figure 5‑14: New diode bridge with crimp connectors secured



Figure 5‑15: 10 mF capacitor bank

### Measurements

The 160W transformer drops about 1V per 1A that is drawn. It is 30V unloaded, and fits within the max power supply rating for the LM358 [1] being 32V. Rseries was also calculated to be below 0.1 ohm.

Under 1Amp load, the Vsteady is 13.7V and Vcc is 13.0V.

### Comparison of theoretical and measured values

With Vcc being 13V and Vsteady being 13.7V, the neat trick in 5.2.3 does not appear to work. This is because the ‘steady’ voltage eventually discharges.

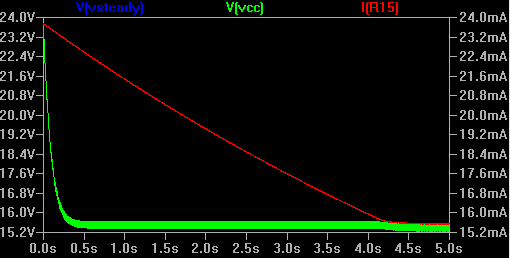


Figure 5‑16: steady voltage over time

See how Vsteady and Vcc end up being 0.7V apart. This was the voltage used for the demo, and would explain a few discrepancies which will be mentioned later on.

### Conclusion and recommendation

A separate transformer would indeed have kept Vsteady much higher. One solution would be to use one of the 9V transformers as a voltage doubler, and to use an 18V Zener (with transistor) to power the op amps and the rest of the circuit.

## Zener Constant-Voltage Reference

### Literature Study

*“Zeners make great fireworks.”*

### Design

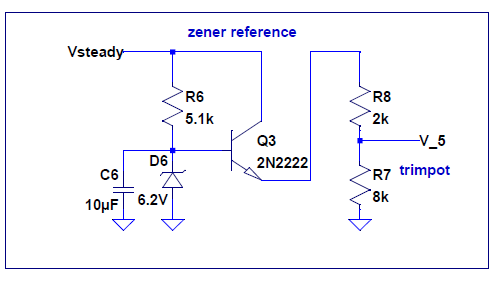


Figure 5‑17: Zener reference circuit

The Zener is biased in the linear region. See appendix for Calculations. C6 is used to stabilise the Zener reference somewhat. Since the transistor draws much less current than the trim pot, it avoids additional potentiometers drawing too much current from the Zener biasing resistor, which, by creating too great a voltage drop across it, would extend under the operating voltage of the Zener, and thus create a mere voltage divider. The trim pot is tweaked to obtain an exact 5V reference. By doing so, one matches with the domain of the Arduino, and thus simplifies relevant calculations. Details are documented in Appendix G.

### Analysis

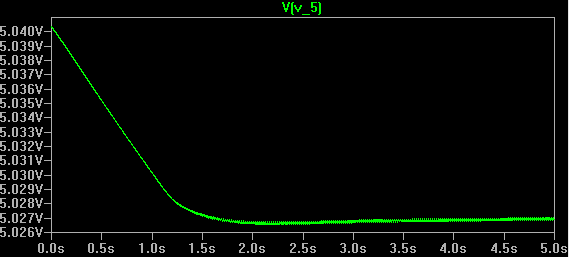


Figure 5‑18: Zener circuit tuned output

The ripple, according to spice, is 187uV.

### Building the circuit

### Measurements

The tuned Zener output is exactly 5V. The ripple is too small to measure on a scope, even on 2mV per division.

### Comparison of theoretical and measured values

There is negligible difference.

### Conclusion and recommendation

The Zener reference is easy to build, especially since it can be tweaked.

There exists an internal 1.1V Zener reference, which should allow more accuracy should it be accounted for in the design of the system. Although it requires two extra op amps to scale the 5V compatible ADC readings down to 1.1V, it means that there does not need to be an 8-12V external supply to the Arduino to power the 5V Zener, which satisfies this design. In the end, powering the 5V Zener is inevitable, since the platform is easily scalable, one might want to add extra features requiring processing power from the microcontroller. See extra features in Appendix G.

### Literature Study

## Outboard Pass transistor stage

### Literature Study

*“Passing engineering is mandatory”.*

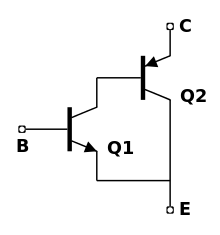


Figure 5‑19: Sziklai pair

The Sziklai pair [2] was considered, however not used since the op amps could only output up to V+ – 1.5. To turn off the transistors completely, one needs to match the voltage of the base of Q1 in Figure 5-4 to the collector of Q2. In the ideal sense, it would have a lower Vce drop as opposed to the Darlington pair.

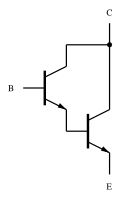


Figure 5‑20: Darlington pair

The Darlington pair was considered since it can be fully turned off. It has a larger collector-emitter voltage drop, however.

### Design

It was decided to use 2N3055 power transistors [3] as they are more easily mounted on a heatsink, compared to the TO-220. Besides that, they absorb more heat.



Figure 5‑21: 2N3055 transistor

Lastly, it was decided that the whole circuit be powered by a steady voltage source, and that most of the power for the load would be provided solely to the collector of the Darlington pair. This way the system would work in true regulator fashion.

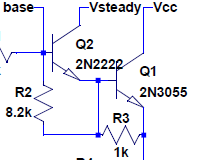


Figure 5‑22: Final pass transistor output stage design

As shown in Fig 5-6, the collector is powered separately to the rest of the circuit.

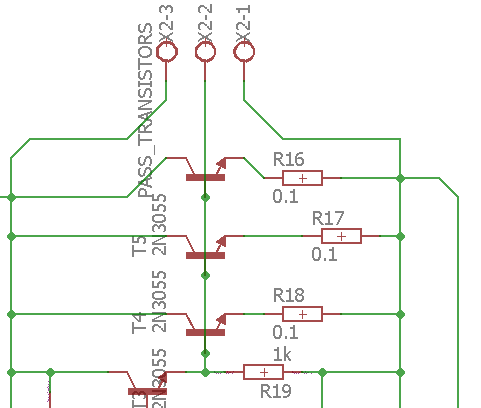


Figure 5‑23: Three outboard pass transistor output stage

Outboard pass transistors are easily scalable, and one can add as many transistors as deemed necessary to dissipate heat. The emitter resistor is required to prevent thermal runaway [4].



Figure 5‑24: Outboard pass transistors to be used in design

### Analysis

The biggest question here is whether the op amp will be able to supply enough current to the Darlington pair.

Assuming 1A flowing through the 2N3055, and with a beta or hfe between 20 and 70 [3] we can assume it is about 50. That means a base current of 20mA. The pre-driver should be more than capable of providing that. Per the 2N2222 datasheet [5], it has a beta value of about 100 under optimal conditions. Therefore, the base current is 200uA. The op amp can provide typically 40mA, therefore the requirement is more than satisfied. Calculating currents when adding multiple outboard pass transistors is simple: merely divide the base and collector currents by the number of such transistors.

Next question is whether the heatsink will handle the current. *See Appendix E.4 for calculations.*

The heatsink can handle x power, which is more than 12W or 160W from either transformer.

### Building the circuit

When the heatsink was built for the 160W transformer, a TIP41C [6] was also used to power the three 2N3055s.

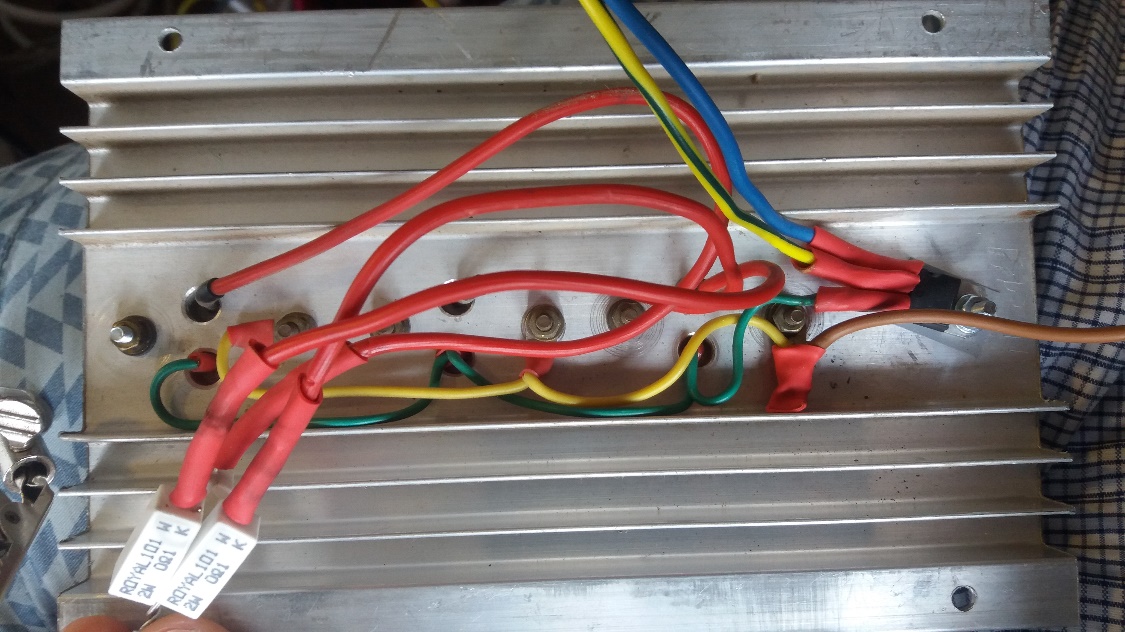


Figure 5‑25: 2N3055 Heatsink

Notice the thermal runaway protection resistors connected to each emitter (white).

The TIP41C ensured that, especially in the case of drawing 19A, that it would source 19/50 = 380mA, and the 2N2222 would draw 3.8mA. The base current of that 2N2222 is now 38uA. It may be thought that this is overkill, but the 2N2222 would get a bit hot if it drew 380mA. This is assuming a beta of 50 for both the TIP41C and the 2N3055, and 100 for the 2N2222. But for the purposes of this paper, it need not be mentioned any further.

### Measurements

Under 1Amp load, the Vsteady is 13.7V and Vcc is 13.0V.

The base of the 2N3055 is 10.900V, the collector matches Vcc and the emitter is 10.1V. The base of the pre-driver 2N2222 is 11.5V. The voltage across the 1k protection resistor is 87mV. That means 87uA is going into the base of the pre-driver.

### Comparison of theoretical and measured values

In the previous section we found out that the base is using 87uA, which is within the range estimated in 5.4.3 being between 38uA and 200uA. 87uA is also very tiny compared to the amount that the op amp can supply, being typically 40mA [1].

### Conclusion and recommendation

The outboard pass transistors are satisfactory.

## Voltage Regulator

### Literature Study

*“Regulations are a chore.”*

Voltage regulators are used in many applications, from power generation, automotive, aerospace down to micro-electronics. It provides a steady power source for sensitive electronics; especially since smoothing capacitors are bulky and expensive, it does not make sense to put such things on small PCBs.

### Design

Figure 5‑26: Voltage regulator

Here is the voltage regulator. Since we know that the base current of the pre-driver is about 38 to 200uA, the voltage drop across R1 is about 3.8 to 200mV. This means that in the worst case scenario, the maximum output voltage is as follows:

Since the transformer drops to about 16V under a 1A load (including capacitors), the op amp can output a max of Vsteady-1.5V. *Assuming that Vsteady and Vcc is the same.* Therefore, the op amp outputs about 14.5V. With a further drop of 200mV, it is 14.3V at the base of the pre-driver. Assuming a drop of 0.6V across Vbe of the pre-driver, and 0.7V across the pass transistor(s), we finally have a max output voltage of 14.3-0.6-0.7=12V. This satisfies the minimum requirements.

R2 and R3 are for more stability due to the differing capacitances in the different driver stages.

### Analysis

### Building the circuit

Originally R4 and R5 had much larger values. We’re talking 27k and 135k. Although it is also a gain of about 6, the current through it was not sufficient to actually output any voltage. Lowering the values (keeping the ratio) fixed the problem.

### Measurements

Under 1Amp load, the Vsteady is 13.7V and Vcc is 13.0V.

The base of the 2N3055 is 10.900V, the collector matches Vcc and the emitter is 10.1V. The base of the pre-driver 2N2222 is 11.5V. The voltage across the 1k protection resistor is 87mV. That means 87uA is going into the base of the pre-driver. This was also found out in 5.4.5.

This means that the op amp is outputting roughly 11.5V. Comparison of theoretical and measured values

### Comparison of theoretical and measured values.

If we had an 18V supply for the op amps, then the max voltage would seem to be, according to measured values, 18 – Vrail-limit – VBE\_driver – VBE\_pre-driver – VR\_sense = 18 – 1.5 – 0.8 – 0.6 - 0.1= 14V

### Conclusion and recommendation

The op amp output, being 11.5V, is 2.2V under 13.7V, which means there is still headroom for more voltage, but not enough that the output voltage will reach 12V under a 1A load. This can easily be rectified, as mentioned in 5.2.7 by having a voltage doubler and a second transformer. In fact, it may even be rectified using the same transformer, but it will need to be investigated.

## Current limiter

### Literature Study

*“Limiting ones imagination is never a good idea”.*

Current limiters are handy to protect external circuits from current surges, shorts, or merely too much current. They protect the power supply as well.

### Design

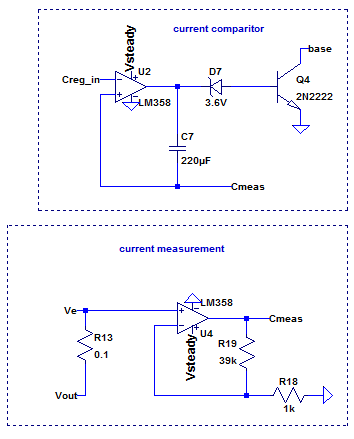


Figure 5‑27: current comparitor and measurement

The current controlling consists of pulling the base voltage of the Darlington pair down to ground. This controls the voltage, which ultimately controls the current.

It was noticed that the comparator could not output a voltage lower than about 3V, so a 3.6V Zener was used to turn off Q4, otherwise the circuit was in current limiting mode when it was suppose to be in constant voltage mode.

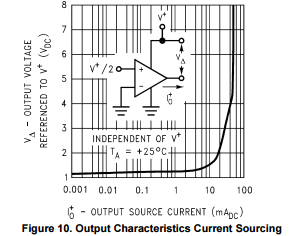
This fact can be verified when looking in the datasheet [1], 

Figure 5‑28: Output characteristics of LM358

That sourcing about 30mA would in fact mean that indeed the lowest output voltage is about 3V. The large current can be attributed to the large capacitive load that it also has to drive.

The large value of C7, 220uF, is used to slow the comparator down. It is basically a large integrator. This means that the steady state error is very low, however, there is overshoot on unit steps such as connecting a load. It also removed oscillations. At least, for functional demonstration purposes, it was satisfactory.

As mentioned in 7.4.2, the measurement op amp merely provides a gain of 40 to the voltage across Rsense. The max voltage across Rsense is 100mV. Therefore, the max voltage that the measurement op amp will provide is 4V. This is within the 5V specifications.

### Analysis

### Building the circuit

### Measurements

### Comparison of theoretical and measured values

### Conclusion and recommendation

# Circuit Integration (Analogue)

## Final System Measurements

Refer to Appendix A for the final measured demonstration results. The power supply had its functionality officially demonstrated in a lab to facilitate non-fabricated readings, and to learn the nature of the power supply.

The designer must interpret the results.

## Interpretation of Results

The readings were done on two channels, and both of which were attached to the output load. The one channel was 5V/div DC coupling and the other was 20mV/div with AC coupling. Unfortunately, poor quality scope leads were used, and 5V per division shows low resolution for smaller voltages.

# Software Design

## Purpose & Requirement

The software interface is required to measure and control the output voltages and currents. As mentioned in 7.3, it is also a platform to add extra functionality. For example, charging batteries.

## Software Literature

## Software Extras

### Control System

A control system was added. Alas, it was not yet merged with the final version at the time of the demo. However, it can still be discussed, here.

The idea was to learn the plant transfer function characteristics of the power supply, without knowing the gain of the voltage regulator or transconductance3 of the current regulator.

It could be described as a PI control system.

Using a known load is preferable, as one can set appropriate step inputs to learn the system.

Let’s assume the case where a 10 ohm calibration load is used.

It gave a step input of a supposed 1000mV, and current greater than 1A. This could be adjusted by the user. Then it would take the average of several voltage measurements.

The user has the option of setting the number of samples before adjusting the output. Samples were sent every 200ms, therefore a good number of samples to test it would be about 5 samples, which means one sees a change every second. It had a slight bit of overshoot as it learned, but it would reach a steady state after about 2-3 seconds.

Upon reaching a steady state within 10% for at least 3 samples, it lowered the current to a specified value, and increased the voltage. For example, voltage now becomes 10V and current becomes 100mA. It learns what the transconductance is.

Upon learning the characteristic plant transfer function of the power supply, it made the voltage gain and transconductance less susceptible to supposed changes in future. This allowed for a smoother steady state, and less overshoot.

### Battery charging profiles

The idea is to be able to easily add a battery charging profile to the software interface and to charge batteries such as NiMH, Pb and Li batteries[[2]](#footnote-2).

To do that one needs to be able to identify the battery. Of course, one can measure the voltage of the battery. For example, a Li cell is usually between 3.7V and 4.2V, a NiMH between 1.2V and 1.4V, and a Pb cell is about 1.9V to 2.1V depending on its charge. It would work well, until those ranges overlap with increasing cell count, then one cannot be positively sure of the cell count. At least, if one knows what kind of battery it is, then one can easily get the cell count. Usually a battery will show its capacity on the outside; for example: 2200mAh, 6Ah, 900mAh etc. Sometimes they give the C rating[[3]](#footnote-3) as well, but for our purposes we will assume the C rating is 1. One could even discharge the battery to determine the C rating, but that risks overheating and explosions if one is not careful.

*In fact, to be very safe, we will charge batteries at 0.1C. It is a generally accepted value as well. The purpose of the project is anyway to demonstrate functionality. In future, once this works, one can investigate fast charging.*

Examples of battery charging profiles are:

Insert here

Therefore, in the end, all one needs to know is the type of battery, and the power supply will do the rest.

### Cellular connectivity

This is an interesting one. The designer would have added this functionality had there been more time. Of course, one can get an estimate of when a battery will finish charging, but since it’s not that easy to determine the exact capacity of a battery upon charging, one might like a notification of when the battery is finished charging, so that one can put the next set on. This is especially helpful if one wants to investigate fast charging, as mentioned in 7.3.2. Sometimes one is busy with time-critical applications that require portable batteries as the sole source of power. Examples include aerial vehicles or electric cars.

## Arduino Interface

The Arduino has set.voltage.value and a set.current.value functions. These functions translate a value into a duty cycle of a 980Hz PWM signal. This signal is then filtered as documented in 7.4.1.

The Arduino also has calibration procedures related to the reading of voltages and currents. This is documented in 7.4.2.

Calculations are documented in Appendix E 12.7.

### Inputs (PWM RC filters)

*This refers to the inputs on the PCB from the Arduino.*

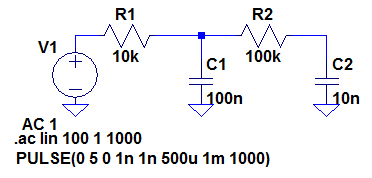


Figure 7‑1: RC filter

The PWM signal from the Arduino needs to be filtered. A second order filter was chosen so that there is less ripple.

Analysis:

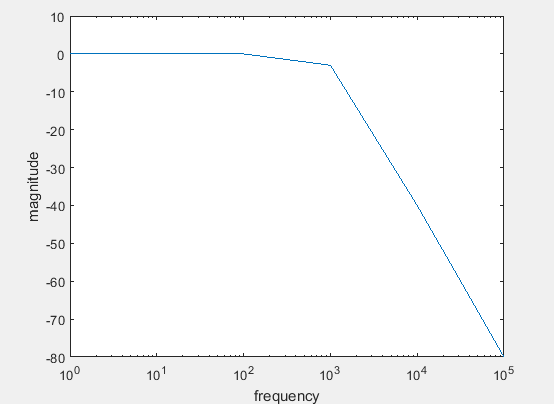


Figure 7‑2: Theoretical 2nd order filter

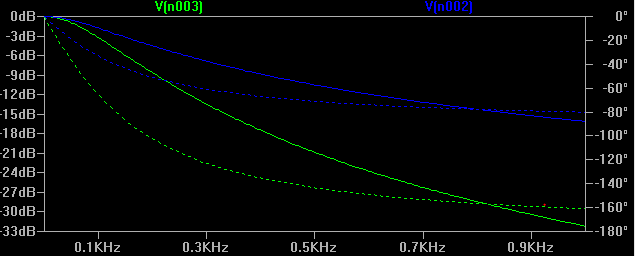


Figure 7‑3: Spice analysis of 2nd order RC filter

Notice how the green line decreases at double the gradient as opposed to the blue line.

Building the Circuit:

The RC filters were attached to a switch, which switched between Arduino input and potentiometer input.

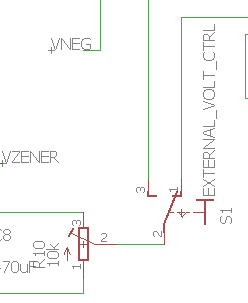


Figure 7‑4: Switch to switch between pot and Arduino

Measurements:

Comparison of theoretical and measured:

Conclusion and recommendations:

The RC filter is quite effective, with next to nothing ripple.

### Outputs

*This refers to the outputs on the PCB to the Arduino.*

Design:



Figure 7‑5: Voltage measuring for Arduino

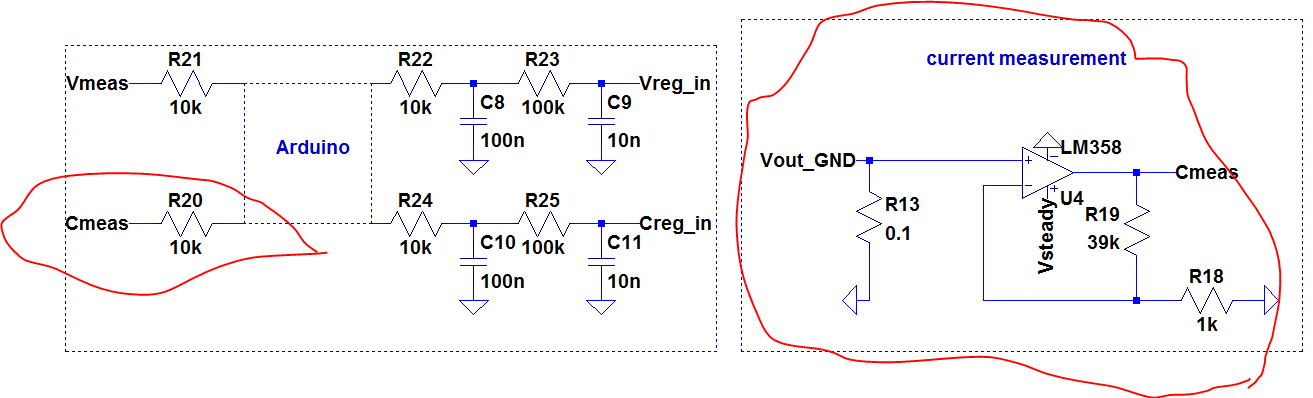


Figure 7‑6: Current measurement

Analysis:

The voltage divider ensures that that maximum voltage that the Arduino will take is 5.1k/(5.1k+22k) \* 26V is 4.89V, which is under 5V.

The current gain ensured that the max measurement to the Arduino is 0.1\*(1+39k/1k) = 0.1\*40 = 4V.

Building the circuit:

As mentioned in the conclusion, protection diodes are thoughtful.

Measurements:

The measurements never exceeded 5V.

Comparison of theoretical and measured values

Conclusion and recommendations:

Adding a 5.1V clamping diode to both the voltage and current measurements would be a good idea. An example follows:

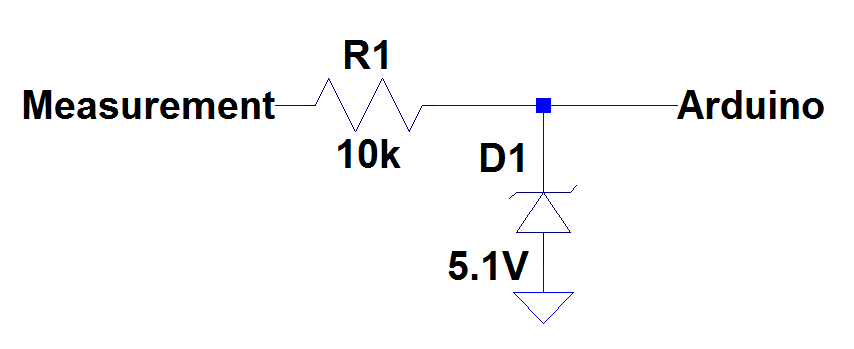


Figure 7‑7: Arduino protection

# Appendix A: Measured Demonstration Results

*See 6.2 for interpretation.*

**Voltage Regulation**

|  |  |  |  |  |  |  |  |
| --- | --- | --- | --- | --- | --- | --- | --- |
| Set Voltage | Set Current | Before Vout DC | Before Vout AC p-p | Load (ohms) | After Vout DC | After Vout AC p-p | Comments |
| 0V | > 1A | 0V | N/A | N/A | N/A | N/A |  |
| >16V | >1A | 16V | N/A | N/A | N/A | N/A |  |
| 1.1V | >1A | 1.2V | <20m | 1.1 | 0.84V | <15m |  |
| 10V | >1A | 9.9V | <20m | 10 | 9.35V | <15m |  |
| 14V | >1A | 14V | <20m | 35 | 13.6V | 110m |  |

**Current Regulation**

|  |  |  |  |  |  |  |  |
| --- | --- | --- | --- | --- | --- | --- | --- |
| 14V | 100mA | 14V | <20m | 10 | 1V | 220m |  |
| 10V | 1A | 10V | 20m | 1.1 | 3.6V | 120m |  |

Extra Hardware functions:

PCB layout + engraving.

Designed for single rail transformer.

**PC controlled**

|  |  |  |  |  |  |  |  |
| --- | --- | --- | --- | --- | --- | --- | --- |
| Set Voltage | Set Current | Load (ohms) | Arduino Voltage | Arduino  Current | After Vout DC | After Vout AC | Comments |
| 10V | 1000mA | 35 | 10.05V | 225mA | 10 | <20m | Scale wrong |
| 10V | 100mA | 35 | 10.05V | 225mA | 10 | <20m | Not working |

# Appendix B: Circuit Diagram

# Appendix C: PCB Layout

Figure 9‑1: PCB design before printing

# Appendix D: Photo of Circuit

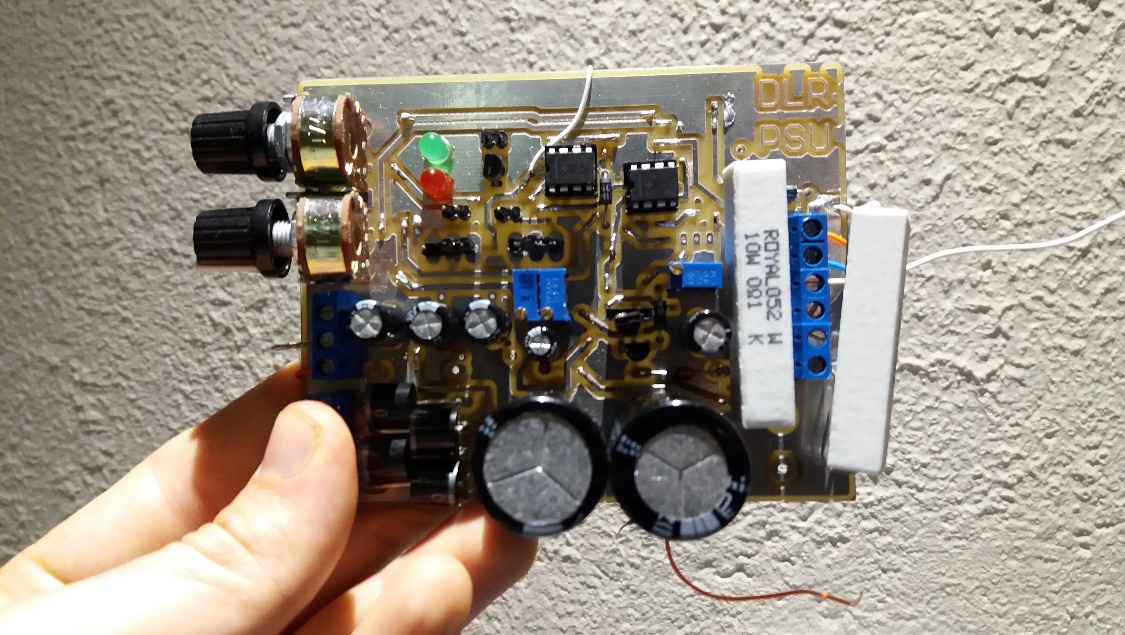


Figure 11‑1: PCB top view

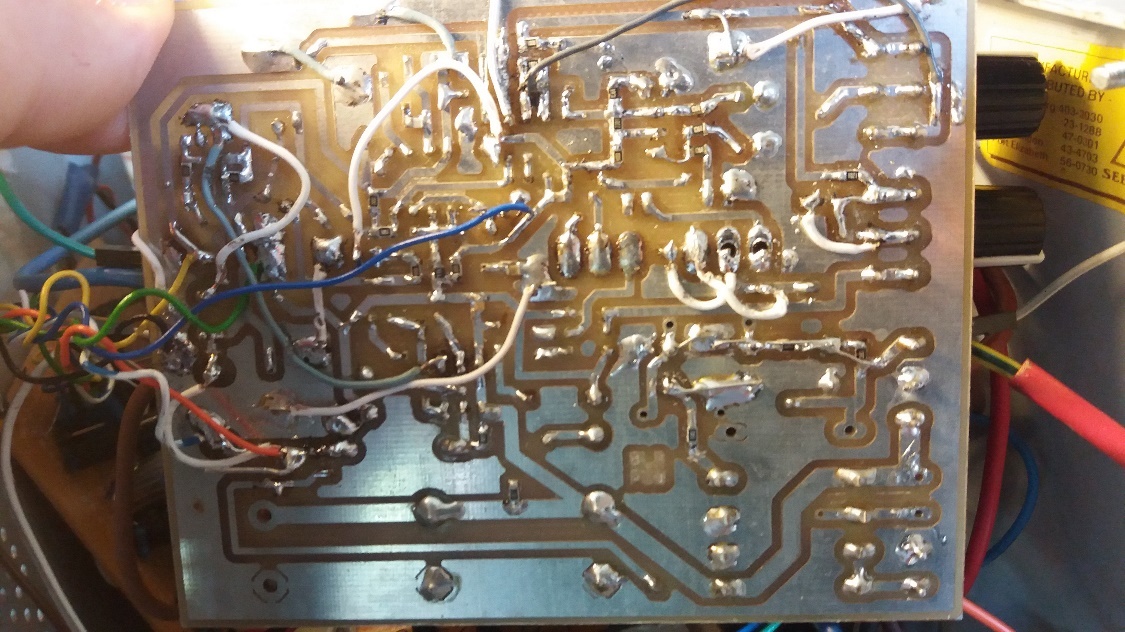


Figure 11‑3: underside of PCB

The underside is SMD mount[[4]](#footnote-4). The wires to the left are connected to a daughterboard which follows.

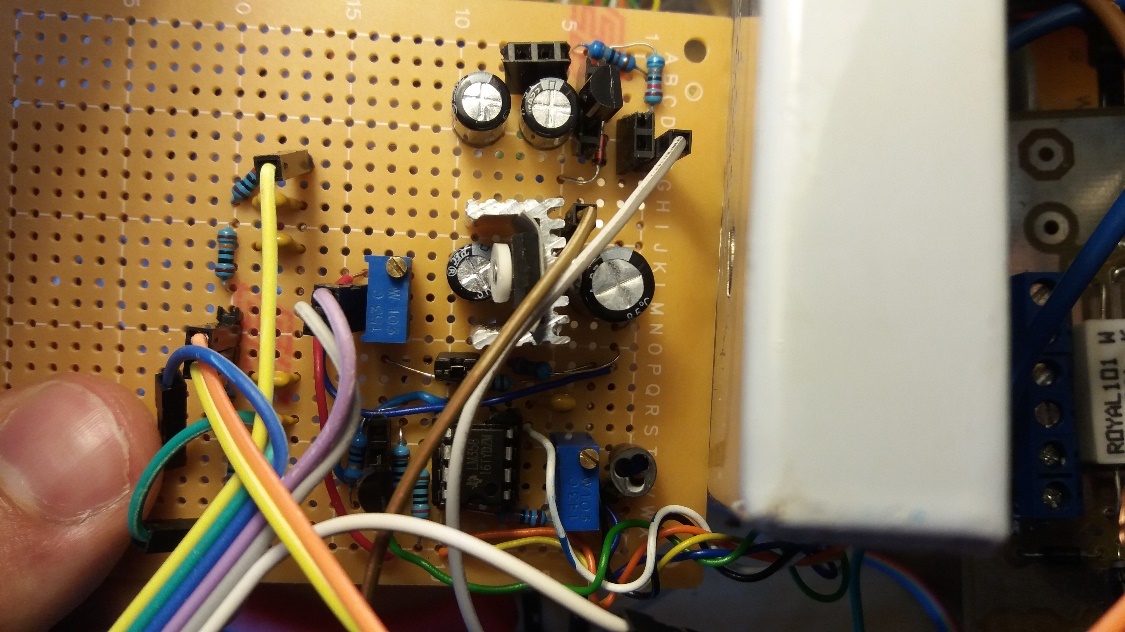


Figure 11‑4: Daughterboard

This daughterboard was only used for high-side extended common-mode current sensing, and the Arduino interface in the end. It also has an unused positive and negative Zener reference.

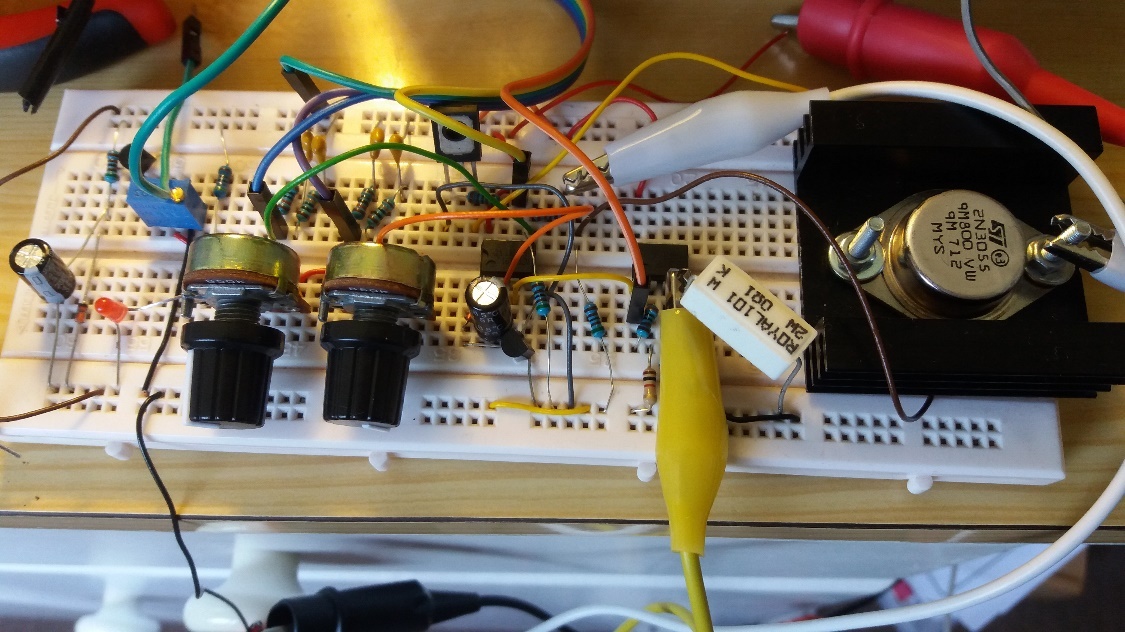


Figure 11‑2: Current circuit breadboarded hours before demo

# Appendix E: Calculations

## Appendix E.1: Transformers

Rseries = (Voc – VL)\*RL/VL

Iload = 22.4/20 = 1.12A

Rseries = (25.6 – 22.4)/1.12 = 2.86 ohms

## Appendix E.2: Rectifiers & Capacitor Banks

## Appendix E.3: Zener Constant Voltage Reference

## Appendix E.4: Pass Output Stage

## Appendix E.5: Voltage Regulator

## Appendix E.6: Current Limiter

## Appendix E.7: Arduino Interface

To determine the v

### PWM filter

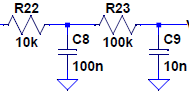


Figure 12‑1: 2nd order PWM filter

Both RC LPFs[[5]](#footnote-5) were chosen to have the same -3dB cut-off frequency.

R22 was chosen to be 10k so that the max current through it would be

Because the -3dB was chosen to be 1ms which is analogous to 1kHz (close to PWM frequency of 980Hz), the capacitor C8 value is:

R23 is chosen to be larger so that the extra current draw through R22 is minimal. Had that not been the case, the max voltage out of the PWM filter would never reach 5V.

Choose R23 = 100k

Since it is now a 2nd order filter, the base 10 logarithmic magnitude decreases at a rate of -40dB/div.

# Appendix F: Source code

"""

@authors:

Jean de Smidt 18393799

Karlien Heyns 18552463

Tristan Nel 18179460

Daniel Leonard Robinson 18361137

This is a program designed to interface between a 15W power supply

and a PC via an Ardiuno.

It uses the serial standard at 9600 baud.

It is for Design E344 at Stellenbosch University

It includes calibration procedures, as well as viewing measurements,

and setting voltages or currents.

It is not yet production level, and is merely a proof of concept

prototype for the purposes of demonstrating the functionality

of a power supply via a PC.

Lastly, this can be used as a platform to add features,

such as battery charging using profiles for lead, NiH, lithium etc.

"""

**import** sys**,** string**,** re

**from** util **import** **\***

**import** PIL**.**Image

**from** PIL **import** **\***

**import** PIL**.**ImageTk

**import** serial**.**tools**.**list\_ports

**if** sys**.**version\_info**<(**3**,**3**,**0**):**

**from** Tkinter **import** **\***

**else:**

**from** tkinter **import** **\***

**import** tkFont

# If you see an error like this:

# File "C:\GIT\software-interface\SerialComms.py", line 12, in <module>

# import serial

# ImportError: No module named serial

# Then run this command in your terminal (CMD)

# python -m pip install pyserial

# If you see an error like this:

# Traceback (most recent call last):

# File "C:\Git repos\software-interface\power-supply-gui.py", line 3, in <module>

# import PIL.Image

# ImportError: No module named PIL.Image

# Then run this command in your terminal (CMD)

# python -m pip install image

**import** SerialComms

#s = SerialComms.SerialComms('COM1',9600)

#s.receive()

root **=** Tk**()**

root**.**title**(**"Bruhhhhh. pls."**)**

root**.**geometry**(**"580x720"**)**

root**.**resizable**(**width**=False,**height**=False)**

frame **=** Frame**(**root**,** width**=**580**,**height**=**720**)**

#frame.configure(width=800,height=800)

frame**.**grid\_propagate**(**0**)**

frame**.**grid**(**row **=** 0**,** column **=** 0**)**

labelfont **=** tkFont**.**Font**(**family **=** "Verdana"**,** size **=** 12**)**

buttonfont **=** tkFont**.**Font**(**family **=** "Verdana"**,** size **=** 11**,** weight **=** tkFont**.**BOLD**)**

image **=** PIL**.**Image**.**open**(**"a.png"**)**

photo **=** PIL**.**ImageTk**.**PhotoImage**(**image**)**

# variables

stream\_enabled **=** **False**

stream\_checked **=** **False**

connected **=** **False**

SerialComm **=** **None**

#calibration variables

#outvalue = vratio\*(vmeasured-voffset)

vratio **=** 1.0

voffset **=** 0.0

iratio **=** 1.0

ioffset **=** 0.0

**print(**"First instance of ratio"**)**

# text fields

stream\_text **=** StringVar**()**

volt\_meas\_val\_text **=** StringVar**()**

curr\_meas\_val\_text **=** StringVar**()**

#functions

**def** on\_off**(**enabled**):**

**if** enabled**:**

**return** "on"

**else:**

**return** "off"

#All the labels

background **=** Label**(**frame**,** image **=** photo**).**place**(**x**=**0**,**y**=**0**,**relwidth**=**1**,** relheight**=**1**)**

voltagelabel **=** Label**(**frame**,** text**=**"Voltage"**,** bg**=**"black"**,** fg **=** "green"**,** font **=** labelfont**)**

currentlabel **=** Label**(**frame**,** text**=**"Current"**,** bg**=**"black"**,** fg **=** "green"**,** font **=** labelfont**)**

comportlabel **=** Label**(**frame**,** text**=**"COM Port"**,** bg**=**"black"**,** fg **=** "green"**,** font **=** labelfont**)**

baudratelabel **=** Label**(**frame**,** text**=** "Baud Rate"**,** bg**=**"black"**,** fg **=** "green"**,** font **=** labelfont**)**

setvreflabel **=** Label**(**frame**,** text**=** "Set Voltage Reference"**,** bg **=** "black"**,** fg **=** "green"**,** font **=** labelfont**)**

setcreflabel **=** Label**(**frame**,** text**=** "Set Current Reference"**,** bg **=** "black"**,** fg **=** "green"**,** font **=** labelfont**)**

errorlabel **=** Label**(**frame**,** text**=**"Disconnected"**,** bg **=** "black"**,** fg**=**"red"**,** font **=** labelfont**)**

otherlabel **=** Label**(**frame**,** text **=** "Other commands"**,** bg **=** "black"**,** fg **=** "green"**,** font **=** labelfont**)**

voltagevaluelabel **=** Label**(**frame**,** textvariable**=**volt\_meas\_val\_text**,** bg **=** "black"**,** fg**=**"white"**,** font **=** labelfont**)**

currentvaluelabel **=** Label**(**frame**,** textvariable**=**curr\_meas\_val\_text**,** bg **=** "black"**,** fg**=**"white"**,** font **=** labelfont**)**

setvoltagelabel **=** Label**(**frame**,** text**=**"Set Voltage"**,** bg**=**"black"**,** fg**=**"white"**,** font**=**labelfont**)**

setcurrentlabel **=** Label**(**frame**,** text**=**"Set Current"**,** bg**=**"black"**,** fg**=**"white"**,** font**=**labelfont**)**

#All the textentries

v**=**StringVar**()**

baudentry **=** Entry**(**frame**)**

comportentry **=** Entry**(**frame**)**

vrefentry **=** Entry**(**frame**)**

crefentry **=** Entry**(**frame**)**

sendentry **=** Entry**(**frame**)**

setvoltageentry **=** Entry**(**frame**)**

setcurrententry **=** Entry**(**frame**)**

setvoltagegainentry **=** Entry**(**frame**)**

setcurrentgainentry **=** Entry**(**frame**)**

#All the buttons

connect **=** Button**(**frame**,** text**=**"Connect"**,** font **=** buttonfont**)**

vcalz **=** Button**(**frame**,** text**=**"Voltage Calibrate zero"**,** font **=** buttonfont**)**

vcalo **=** Button**(**frame**,**text **=** "Voltage Calibrate other"**,** font **=** buttonfont**)**

ccalz **=** Button**(**frame**,**text **=** "Current Calibrate zero"**,** font **=** buttonfont**)**

ccalo **=** Button**(**frame**,**text **=** "Current Calibrate other"**,** font **=** buttonfont**)**

refresh **=** Button**(**frame**,** text **=** "Refresh"**,** font **=** buttonfont**)**

send **=** Button**(**frame**,** text **=** "Send"**,** bg**=**"dark green"**,** fg**=**"white"**,** font **=** buttonfont**)**

clear **=** Button**(**frame**,** text **=** "Clear"**,** bg**=**"dark blue"**,** fg**=**"white"**,** font **=** buttonfont**)**

streambutton **=** Button**(**frame**,** textvariable**=**stream\_text**,** font **=** labelfont**)**

setvoltagebutton **=** Button**(**frame**,** text**=**"Set Voltage"**,** font**=**labelfont**)**

setcurrentbutton **=** Button**(**frame**,** text**=**"Set Current"**,** font**=**labelfont**)**

#All the label layout

frame**.**grid\_columnconfigure**(**0**,** minsize **=** 150**)**

frame**.**grid\_columnconfigure**(**1**,** minsize **=** 100**)**

frame**.**grid\_rowconfigure**(**0**,** minsize **=** 50**)**

frame**.**grid\_rowconfigure**(**1**,** minsize **=** 50**)**

frame**.**grid\_rowconfigure**(**2**,** minsize **=** 50**)**

frame**.**grid\_rowconfigure**(**3**,** minsize **=** 50**)**

frame**.**grid\_rowconfigure**(**4**,** minsize **=** 50**)**

frame**.**grid\_rowconfigure**(**6**,** minsize **=** 50**)**

frame**.**grid\_rowconfigure**(**9**,** minsize **=** 50**)**

frame**.**grid\_rowconfigure**(**13**,** minsize **=** 50**)**

frame**.**grid\_rowconfigure**(**14**,** minsize **=** 50**)**

voltagelabel**.**grid**(**row **=**0**,** column**=**0**,** sticky**=**E**)**

voltagevaluelabel**.**grid**(**row**=**0**,** column**=**2**,** sticky**=**W**+**E**)**

currentlabel**.**grid**(**row**=**1**,**column**=**0**,**sticky**=**E**)**

currentvaluelabel**.**grid**(**row**=**1**,** column**=**2**,** sticky**=**E**)**

comportlabel**.**grid**(**row**=**2**,**column**=**0**,**sticky**=**E**)**

baudratelabel**.**grid**(**row**=**3**,**column**=**0**,**sticky**=**E**)**

errorlabel**.**grid**(**row**=**4**,** column**=**0**,** sticky**=**E**)**

setvreflabel**.**grid**(**row**=**6**,**column**=**0**,**sticky**=**E**)**

setcreflabel**.**grid**(**row**=**9**,**column**=**0**,**sticky**=**E**)**

otherlabel**.**grid**(**row**=**15**,**column**=**0**,**sticky**=**E**)**

#All the textentry layout

baudentry**.**grid**(**row**=**3**,**column**=**1**,**columnspan**=**2**,** sticky**=**E**)**

comportentry**.**grid**(**row**=**2**,**column**=**1**,**columnspan**=**2**,** sticky**=**E**)**

vrefentry**.**grid**(**row**=**6**,**column**=**1**,**columnspan**=**2**,** sticky**=**E**)**

crefentry**.**grid**(**row**=**9**,**column**=**1**,**columnspan**=**2**,** sticky**=**E**)**

sendentry**.**grid**(**row**=**15**,**column**=**1**,** columnspan**=**2**,** sticky**=**E**)**

setvoltagegainentry**.**grid**(**row**=**13**,** column**=**1**,** columnspan**=**1**,** sticky**=**E**)**

setvoltageentry**.**grid**(**row**=**13**,** column**=**2**,** columnspan**=**1**,** sticky**=**E**)**

setcurrentgainentry**.**grid**(**row**=**14**,** column**=**1**,** columnspan**=**1**,** sticky**=**E**)**

setcurrententry**.**grid**(**row**=**14**,** column**=**2**,** columnspan**=**1**,** sticky**=**E**)**

# maybe we don't need these two lines?

#frame.columnconfigure(2, weight=3)

#frame.columnconfigure(3, weight=3)

#All the buttons layout

connect**.**grid**(**row**=**4**,**column**=**1**,**sticky **=** E**)**

refresh**.**grid**(**row**=**4**,**column**=**2**,**sticky**=**E**)**

setvreflabel**.**grid**(**row**=**6**,**column**=**0**)**

vcalz**.**grid**(**row**=**7**,**column**=**0**,**sticky**=**E**)**

vcalo**.**grid**(**row**=**8**,**column**=**0**,**stick**=**E**)**

ccalz**.**grid**(**row**=**10**,**column**=**0**,**sticky**=**E**)**

ccalo**.**grid**(**row**=**11**,**column**=**0**,**sticky**=**E**)**

streambutton**.**grid**(**row**=**12**,**column**=**1**,**sticky**=**E**)**

clear**.**grid**(**row**=**15**,**column**=**1**,**sticky**=**E**)**

send**.**grid**(**row**=**15**,**column**=**2**,**sticky**=**E**)**

setvoltagebutton**.**grid**(**row**=**13**,** column**=**0**,** sticky**=**E**)**

setcurrentbutton**.**grid**(**row**=**14**,** column**=**0**,** sticky**=**E**)**

# setup

stream\_text**.**set**(**"stream is %s" **%** on\_off**(False))** #maybe change text to a command ("Turn stream on") to show it is a button that can be pressed

volt\_meas\_val\_text**.**set**(**"0.000 V"**)** #should always display the same nr of digits

curr\_meas\_val\_text**.**set**(**"0.000 A"**)**

**def** find\_arduino**():**

serPort **=** ""

# Find Live Ports

ports **=** list**(**serial**.**tools**.**list\_ports**.**comports**())**

**for** p **in** ports**:**

# debug automatically finding Arduino by uncommenting next line

# print p

**for** obj **in** p**:**

**if** "Arduino" **in** obj**:**

serPort **=** p**[**0**]**

**print** "Found Arduino at %s\n" **%** serPort

**return** serPort

#All the events

**def** connectf**(**event**):**

**global** SerialComm

**global** connected

#Check if existing connection is open

**if** SerialComm **is** **None:** #Open new connection

**print** 'Connecting'

**try:**

#Get values

baud **=** baudentry**.**get**()**

comport **=** comportentry**.**get**()**

#Input error checking

check **=** **False**

r **=** re**.**compile**(**'^COM\d+$'**)**

# print r.match(comport)

**if** **not** baud**.**isdigit**():**

check **=** **True**

baudentry**.**delete**(**0**,**END**)**

baudentry**.**insert**(**0**,**'Enter a number as the baud rate!'**)**

**if** **not** r**.**match**(**comport**)** **is** **not** **None:**

check **=** **True**

comportentry**.**delete**(**0**,**END**)**

comportentry**.**insert**(**0**,**'Must be format: COM%number%'**)**

**if** check**:**

errorlabel**.**config**(**text**=**'Invalid input!'**,** fg**=**'Red'**)**

**return**

errorlabel**.**config**(**text**=**''**)**

#Establish connection

SerialComm **=** SerialComms**.**SerialComms**(**comport**,** baud**)**

SerialComm**.**open**()**

#Change UI

connect**.**config**(**text**=**'Disconnect'**)**

errorlabel**.**config**(**text**=**'Connected'**,** fg**=**'Green'**,** font **=** labelfont**)**

connected **=** **True**

**print** "Connected!"

**except:**

SerialComm **=** **None**

errorlabel**.**config**(**text**=**'Unable to connect! Try again'**,** fg**=**'Red'**)**

**return**

**else:** #Disconnect

connected **=** **False**

**try:**

**print** 'Disconnecting...'

#Close connection

SerialComm**.**close**()**

SerialComm **=** **None**

#Change GUI

connect**.**config**(**text**=**'Connect'**)**

errorlabel**.**config**(**text**=**'Disconnected'**,** fg**=**'Red'**,** font **=** labelfont**)**

**except:**

errorlabel**.**config**(**text**=**'Unable to disconnect! Try again'**)**

**return**

**def** vcalzerof**(**event**):**

setVOffset**()**

**def** vcalotherf**(**event**):**

**try:**

x**=**float**(**vrefentry**.**get**())**

setVRatio**()**

**except** ValueError**:**

**print(**"You must enter a number "**)**

**def** ccalzerof**(**event**):**

setIOffset**()**

**def** ccalotherf**(**event**):**

**try:**

x**=**float**(**crefentry**.**get**())**

setIRatio**()**

**except** ValueError**:**

**print(**"You must enter a number "**)**

**def** sendf**(**event**=**0**):**

command **=** sendentry**.**get**()**

SerialComm**.**send**(**command**)**

**print** command

**def** clearf**(**event**=**0**):**

sendentry**.**delete**(**0**,**END**)**

**print** "cleared"

**def** streamf**(**event**=**0**):**

**global** stream\_enabled

stream\_enabled **^=** **True** #should change to false with new button text?

**print(**"streaming measurements: " **+** str**(**stream\_enabled**))**

stream\_text**.**set**(**"stream is %s" **%** on\_off**(**stream\_enabled**))** #("Turn stream on/off")

SerialComm**.**send**(**"stream.%s" **%** on\_off**(**stream\_enabled**))**

def refresh\_comport(event=0):

comportentry.delete(0,END)

comportentry.insert(0,find\_arduino())

def setvoltagef(event=0):

global SerialComm

global connected

if connected:

try:

# Get input data

gainText = setvoltagegainentry.get()

voltageText = setvoltageentry.get()

## CHECK THE INPUT

gain = float(gainText)

voltage = float(voltageText)

# Calculate command value

value = int(voltage/gain/5\*25600)

# Send command

cmd = "set.voltage." + str(value)

print cmd

SerialComm.send(cmd)

except IndexError:

pass

def setcurrentf(event=0):

global SerialComm

global connected

if connected:

try:

# Get input data

gainText = setcurrentgainentry.get()

currentText = setcurrententry.get()

## CHECK THE INPUT

gain = float(gainText)

current = float(currentText)

# Calculate command value

value = int(current/gain/5\*1280)

# Send command

cmd = "set.current." + str(value)

print cmd

SerialComm.send(cmd)

except IndexError:

pass

#Button bindings to the events

connect.bind("<Button-1>",connectf)

refresh.bind("<Button-1>",refresh\_comport)

vcalz.bind("<Button-1>",vcalzerof)

vcalo.bind("<Button-1>",vcalotherf)

ccalz.bind("<Button-1>",ccalzerof)

ccalo.bind("<Button-1>",ccalotherf)

send.bind("<Button-1>",sendf)

clear.bind("<Button-1>",clearf)

streambutton.bind("<Button-1>",streamf)

setvoltagebutton.bind("<Button-1>",setvoltagef)

setcurrentbutton.bind("<Button-1>",setcurrentf)

update\_ms = 99

def updateMeasurements():

global connected

global stream\_checked

global vratio

global voffset

global iratio

global ioffset

if connected:

stream = SerialComm.receive()

if (len(stream) > 0):

if not stream\_checked:

# Check if there is a stream of measurements and

# update stream button

stream\_checked=True

stream\_enabled=False

streamf()

try:

# Process stream of voltages and currents from Arduino

current\_string = stream[-1]

voltage\_string = stream[-2]

if isValidCurrent(current\_string):

current\_raw = extractCurrent(current\_string)

curr\_meas=current\_raw;

curr\_meas\_val\_text.set("%d mA" % (iratio\*(curr\_meas-ioffset)))

if isValidVoltage(voltage\_string):

voltage\_raw = extractVoltage(voltage\_string)

volt\_meas=float(voltage\_raw)/1000;

#print("What is the ratio %f", vratio)

#print("What is the offset %f", voffset)

#print("What is the Meas %f", volt\_meas)

volt\_meas\_val\_text.set("%.3f V" % (vratio\*(volt\_meas-voffset)))

except IndexError:

pass

root.after(update\_ms,updateMeasurements)

def setVRatio():

global connected

global stream\_checked

global vratio

global voffset

if connected:

stream = SerialComm.receive()

if (len(stream) > 0):

if not stream\_checked:

# Check if there is a stream of measurements and

# update stream button

stream\_checked=True

stream\_enabled=False

streamf()

try:

# Process stream of voltages and currents from Arduino

voltage\_string = stream[-2]

if isValidVoltage(voltage\_string):

voltage\_raw = extractVoltage(voltage\_string)

volt\_meas=float(voltage\_raw)/1000;

vratio = float(vrefentry.get())/volt\_meas;

print("Actual: %s",vrefentry.get())

print("Measured: %s",volt\_meas)

print("Changed Ratio %f", vratio)

volt\_meas\_val\_text.set("%.3f V"% (vratio\*(volt\_meas-voffset)))

except IndexError:

pass

def setVOffset():

global connected

global stream\_checked

global voffset

global vratio

if connected:

stream = SerialComm.receive()

if (len(stream) > 0):

if not stream\_checked:

# Check if there is a stream of measurements and

# update stream button

stream\_checked=True

stream\_enabled=False

streamf()

try:

# Process stream of voltages and currents from Arduino

voltage\_string = stream[-2]

if isValidVoltage(voltage\_string):

voltage\_raw = extractVoltage(voltage\_string)

volt\_meas=float(voltage\_raw)/1000;

voffset = volt\_meas;

print("Voffset is %f",voffset)

volt\_meas\_val\_text.set("%.3f V" % (vratio\*(volt\_meas-voffset)))

except IndexError:

pass

def setIRatio():

global connected

global stream\_checked

global iratio

global ioffset

if connected:

stream = SerialComm.receive()

if (len(stream) > 0):

if not stream\_checked:

# Check if there is a stream of measurements and

# update stream button

stream\_checked=True

stream\_enabled=False

streamf()

try:

# Process stream of voltages and currents from Arduino

current\_string = stream[-1]

if isValidCurrent(current\_string):

current\_raw = extractCurrent(current\_string)

curr\_meas=current\_raw;

ioffset = curr\_meas

print("I offset %f",ioffset)

curr\_meas\_val\_text.set("%d mA" % (iratio\*(curr\_meas-ioffset)))

except IndexError:

pass

def setIOffset():

global connected

global stream\_checked

global iratio

global ioffset

if connected:

stream = SerialComm.receive()

if (len(stream) > 0):

if not stream\_checked:

# Check if there is a stream of measurements and

# update stream button

stream\_checked=True

stream\_enabled=False

streamf()

try:

# Process stream of voltages and currents from Arduino

current\_string = stream[-1]

if isValidCurrent(current\_string):

current\_raw = extractCurrent(current\_string)

curr\_meas=current\_raw;

ioffset = curr\_meas

curr\_meas\_val\_text.set("%d mA" % (iratio\*(curr\_meas-ioffset)))

except IndexError:

pass

def setup():

# runs once on startup

refresh\_comport()

baudentry.delete(0,END)

baudentry.insert(0,9600)

try:

setup()

root.after(update\_ms,updateMeasurements)

root.mainloop()

except KeyboardInterrupt:

print 'Goodbye'

finally:

if SerialComm is not None:

print 'Disconnecting...'

try:

SerialComm.close()

except:

errorlabel.config(text="Failed to disconnect")

exit

# Appendix G: Extra information

Knowing the exact gain is useful to calibrate the software interface. The interface needs to know what voltage out of the PWM RC filter will result in the desired output. The exact gain can be different due to ground feedback (mentioned in 5.3.3), especially when measuring currents at the preamplification stage. Here are at least two methods to determine it.

## Voltage Gain

To determine the voltage gain of the regulator, a multimeter is required. Since it was mentioned in 5.3.2 Zener Design that we should match the voltage domain of the Arduino, we can utilise the full 5V range from the Arduino (determined by PWM and RC filter, and ultimately frequency), which is as accurate as one can get.

To determine the voltage gain, set the middle pin of the voltage pot to 1V, and measure the output voltage. The output voltage will match the voltage gain.

## Current Gain (transconductance)

A similar method as described in 14.1 can be used to determine the exact current gain. Measure 1V on the middle pin of the current adjust potentiometer with a known load 10 ohms or higher, and measure the output voltage to determine the output current.

## Extra Hardware Features

### Battery charging profiles

See 7.3.2 for more information.

Switches can be added to identify between NiMH, Pb and Li cells.

# Figures

[5]

[Figure 5‑1: Bridge and capacitors 5-12](#_Toc465022951)

[Figure 5‑2: Zener reference circuit 5-13](#_Toc465022952)

[Figure 5‑4: Sziklai pair 5-15](#_Toc465022953)

[Figure 5‑3: Darlington pair 5-15](#_Toc465022954)

[Figure 5‑5: 2N3055 transistor 5-16](#_Toc465022955)

[Figure 5‑6: Final pass transistor output stage design 5-16](#_Toc465022956)

[Figure 5‑7: Three outboard pass transistor output stage 5-17](#_Toc465022957)

[Figure 5‑8: RC filter 5-20](#_Toc465022958)

[Figure 5‑9: Spice analysis of 2nd order RC filter 5-20](#_Toc465022959)

[Figure 5‑10: Switch to switch between pot and arduino 5-21](#_Toc465022960)

[Figure 5‑11: Voltage measuring for Arduino 5-22](#_Toc465022961)

[Figure 5‑12: Current measurement 5-22](#_Toc465022962)

[Figure 5‑13: Arduino protection 5-23](#_Toc465022963)

[Figure 9‑1: PCB design before printing 10-32](file:///C:\GIT\digital-power-supply\wiki\Edesign344Project.docx#_Toc465022964)

[Figure 12‑1: 2nd order PWM filter 12-34](#_Toc465022965)

# Glossary

Transconductance.

C rating – It is a multiple of the capacity, and thus reflective of the rate at which one can charge/discharge the battery.

# Index

SMD mount, 11-41

transconductance, 7-32

# Bibliography

|  |  |
| --- | --- |
| [1] | “LM358,” [Online]. Available: www.ti.com/lit/ds/symlink/lm158-n.pdf. |
| [2] | “https://en.wikipedia.org/wiki/Sziklai\_pair,” [Online]. |
| [3] | “https://en.wikipedia.org/wiki/2N3055,” [Online]. |
| [4] | “https://en.wikipedia.org/wiki/Thermal\_runaway,” [Online]. |
| [5] | D. Jones. [Online]. Available: https://youtu.be/tF2krfxYc68. |

http://www.electronics-lab.com/project/0-30-vdc-stabilized-power-supply-with-current-control-0-002-3-a/

http://diyfan.blogspot.co.za/2013/03/adjustable-lab-power-supply-take-two.html

http://www.eleccircuit.com/cheap-adjustable-0-30v-2a-laboratory-dc-power-supply/

http://www.repeater-builder.com/astron/pix/astron-rs10a-1981-09-01.jpg

http://www.electro-tech-online.com/threads/variable-voltage-regulator-using-lm723.116977/

http://www.sentex.ca/~mec1995/circ/ps3010/ps3010a.html

http://www.northcountryradio.com/PDFs/030702002.pdf

http://www.tuxgraphics.org/electronics/201005/bench-power-supply-v3.pdf

http://www.avdweb.nl/arduino/libraries/fast-pwm-dac.html

http://shop.tuxgraphics.org/electronic/detail\_microcontroller\_powersupply.html

https://github.com/tschutter/digital-dc-power-supply

http://www.tuxgraphics.org/common/src2/article10051/

http://stackoverflow.com/questions/16058695/arduino-nano-timers

http://electronicdesign.com/power/whats-all-error-budget-stuff-anyhow

http://www.ti.com/lit/ds/symlink/lm158-n.pdf

http://www.electro-tech-online.com/threads/ltsspice-question.29885/

http://homepages.which.net/~paul.hills/Circuits/Spice/ModelIndex.html

http://www.audio-perfection.com/spice-ltspice/distortion-measurements-with-ltspice.html

http://www.linear.com/solutions/5788

http://www.ti.com/lit/an/slyt183/slyt183.pdf

http://www.simonbramble.co.uk/lt\_spice/ltspice\_lt\_spice\_tutorial\_3.htm

http://www.eetimes.com/document.asp?doc\_id=1279404

http://www.ti.com/lit/ml/slyb194a/slyb194a.pdf

1. Single rail mode means a supply voltage of 0 to Vcc, not -Vcc/2 to Vcc/2 [↑](#footnote-ref-1)
2. See 1 List of Abbreviations [↑](#footnote-ref-2)
3. See Glossary [↑](#footnote-ref-3)
4. See 1 List of Abbreviations [↑](#footnote-ref-4)
5. See 1 List of Abbreviations [↑](#footnote-ref-5)