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, according to 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.

Table of Contents

[Declaration 1-1](#_Toc464946856)

[Abstract 1-2](#_Toc464946857)

[1 List of Abbreviations 1-6](#_Toc464946858)

[2 Introduction 2-7](#_Toc464946859)

[3 Problem Statement 3-8](#_Toc464946860)

[4 Literature Study: System 4-9](#_Toc464946861)

[5 System Design 5-9](#_Toc464946862)

[5.1 Transformers 5-9](#_Toc464946863)

[5.1.1 Literature Study 5-9](#_Toc464946864)

[5.1.2 Design 5-9](#_Toc464946865)

[5.1.3 Analysis 5-10](#_Toc464946866)

[5.1.4 Building the circuit 5-10](#_Toc464946867)

[5.1.5 Measurements 5-10](#_Toc464946868)

[5.1.6 Comparison of theoretical and measured values 5-10](#_Toc464946869)

[5.1.7 Conclusion and recommendations 5-10](#_Toc464946870)

[5.2 Rectifiers & Capacitor Banks 5-11](#_Toc464946871)

[5.2.1 Literature Study 5-11](#_Toc464946872)

[5.2.2 Design 5-11](#_Toc464946873)

[5.2.3 Analysis 5-12](#_Toc464946874)

[5.2.4 Building the circuit 5-12](#_Toc464946875)

[5.2.5 Measurements 5-12](#_Toc464946876)

[5.2.6 Comparison of theoretical and measured values 5-12](#_Toc464946877)

[5.2.7 Conclusion and recommendation 5-12](#_Toc464946878)

[5.3 Zener Constant-Voltage Reference 5-12](#_Toc464946879)

[5.3.1 Literature Study 5-12](#_Toc464946880)

[5.3.2 Design 5-12](#_Toc464946881)

[5.3.3 Analysis 5-13](#_Toc464946882)

[5.3.4 Building the circuit 5-13](#_Toc464946883)

[5.3.5 Measurements 5-13](#_Toc464946884)

[5.3.6 Comparison of theoretical and measured values 5-13](#_Toc464946885)

[5.3.7 Conclusion and recommendation 5-13](#_Toc464946886)

[Literature Study 5-13](#_Toc464946887)

[5.4 Pass Output Stage 5-14](#_Toc464946888)

[5.4.1 Literature Study 5-14](#_Toc464946889)

[5.4.2 Design 5-14](#_Toc464946890)

[5.4.3 Analysis 5-14](#_Toc464946891)

[5.4.4 Building the circuit 5-14](#_Toc464946892)

[5.4.5 Measurements 5-14](#_Toc464946893)

[5.4.6 Comparison of theoretical and measured values 5-14](#_Toc464946894)

[5.4.7 Conclusion and recommendation 5-14](#_Toc464946895)

[5.5 Voltage Regulator 5-15](#_Toc464946896)

[5.5.1 Literature Study 5-15](#_Toc464946897)

[5.5.2 Design 5-15](#_Toc464946898)

[5.5.3 Analysis 5-15](#_Toc464946899)

[5.5.4 Building the circuit 5-15](#_Toc464946900)

[5.5.5 Measurements 5-15](#_Toc464946901)

[5.5.6 Comparison of theoretical and measured values 5-15](#_Toc464946902)

[5.5.7 Conclusion and recommendation 5-15](#_Toc464946903)

[5.6 Current limiter 5-16](#_Toc464946904)

[5.6.1 Literature Study 5-16](#_Toc464946905)

[5.6.2 Design 5-16](#_Toc464946906)

[5.6.3 Analysis 5-16](#_Toc464946907)

[5.6.4 Building the circuit 5-16](#_Toc464946908)

[5.6.5 Measurements 5-16](#_Toc464946909)

[5.6.6 Comparison of theoretical and measured values 5-16](#_Toc464946910)

[5.6.7 Conclusion and recommendation 5-16](#_Toc464946911)

[5.7 Arduino Interface 5-17](#_Toc464946912)

[5.7.1 Inputs (PWM RC filters) 5-17](#_Toc464946913)

[5.7.2 Outputs 5-18](#_Toc464946914)

[6 Circuit Integration (Analogue) 6-19](#_Toc464946915)

[6.1 Final System Measurements 6-19](#_Toc464946916)

[6.2 Interpretation of Results 6-20](#_Toc464946917)

[7 Software Design 7-21](#_Toc464946918)

[7.1 Purpose & Requirement 7-21](#_Toc464946919)

[7.2 Software Literature 7-22](#_Toc464946920)

[7.3 Software Extras 7-23](#_Toc464946921)

[7.3.1 Control System 7-23](#_Toc464946922)

[8 Appendix A: Measured Demonstration Results 8-24](#_Toc464946923)

[9 Appendix B: Circuit Diagram 9-25](#_Toc464946924)

[10 Appendix C: PCB Layout 10-26](#_Toc464946925)

[11 Appendix D: Photo of Circuit 11-27](#_Toc464946926)

[12 Appendix E: Calculations 12-28](#_Toc464946927)

[12.1 Appendix E.1: transformers 12-28](#_Toc464946928)

[13 Appendix F: Source code 13-29](#_Toc464946929)

[14 Appendix G: Extra information 14-30](#_Toc464946930)

[15 Figures 15-31](#_Toc464946931)

[16 Bibliography 16-32](#_Toc464946932)

# List of Abbreviations

* PCB – Printed Circuit Board
* SMPS – Switched Mode Power Supplies
* NiMH – Nickel-Metal Hydride
* Pb – Lead
* Li - Lithium

# 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 (eg. LM317). There are provided components, 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 votlages/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 voltgage after the diode bridge.
* Non-linear effects of op amps, transistors, zeners etc have to be taken into account

# 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 comparsons]

# System Design

This will include the design, analysis, measurements and comparations of theory and results.

## 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 transfomers that the designer had available: a 16V dual secondary (unknown current rating), and a 20V 8A secondary transformer.

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.

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.

However, the greatest one of them all is the 20V 8A one. In future it will be referred to as the 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 actually requires only 1 rail to fully make use of the extra voltage overhead.

### Analysis

### Building the circuit

### Measurements

The 160W transformer drops about 1V per 1A that is drawn. It is 30V unloaded, and fits withing the max power supply rating for the LM358 [1]

### 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.

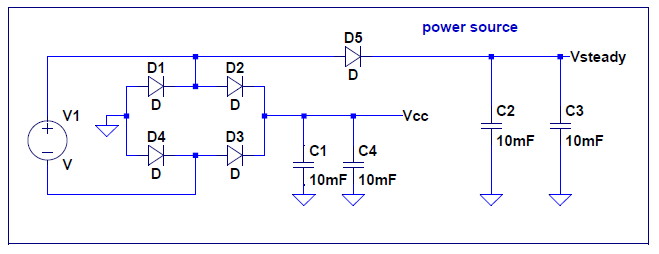


Figure 5‑1: Bridge and capacitors

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

### Building the circuit

### Measurements

### Comparison of theoretical and measured values

### Conclusion and recommendation

## Zener Constant-Voltage Reference

### Literature Study

*“Zeners make great fireworks.”*

### Design

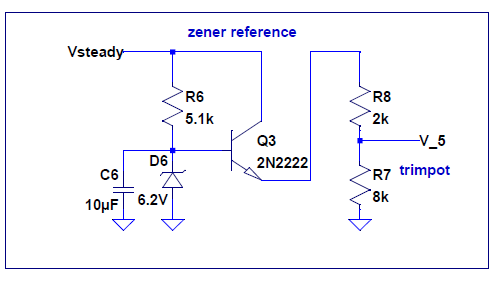


Figure 5‑2: Zener reference circuit

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 trimpot 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

### Building the circuit

### Measurements

### Comparison of theoretical and measured values

### Conclusion and recommendation

There exists an internal 1.1V Zener reference, which should allow a greater degree of 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 Zene, 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

Design

Analysis

Building the circuit

Measurements

Comparison of theoretical and measured values

Conclusion and recommendation

## Pass Output Stage

### Literature Study

*“Passing engineering is mandatory”*

### Design

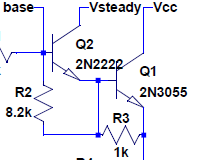


Figure 3: This is the current pass transistor output stage

### Analysis

### Building the circuit

### Measurements

### Comparison of theoretical and measured values

### Conclusion and recommendation

## Voltage Regulator

### Literature Study

### Design

### Analysis

### Building the circuit

### Measurements

### Comparison of theoretical and measured values

### Conclusion and recommendation

## Current limiter

### Literature Study

### Design

### Analysis

### Building the circuit

### Measurements

### Comparison of theoretical and measured values

### Conclusion and recommendation

## 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 5.7.1.

The Arduino also has calibration procedures related to the reading of voltages and currents. This is documented in 5.7.2.

Calculations are documented in Appendix 12.7.

### Inputs (PWM RC filters)

This referes to the inputs on the PCB from the Arduino.

### Outputs

This referes to the outputs on the PCB to the Arduino.

# 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

## 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 transconductance 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.

Lets 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 a number of 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 it’s 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 it’s 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.

# 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

# Appendix E: Calculations

## Appendix E.1: transformers

Rseries

## Appendix E.2: Rectifiers & Capacitor Banks

## Appendix E.3: Zener Constant Voltage Reference

## Pass Output Stage

## Voltage Regulator

## Current Limiter

## Arduino Interface

To determine the v

# 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

[2]

[Figure 5‑1: Bridge and capacitors 5-12](#_Toc464984166)

[Figure 5‑2: Zener reference circuit 5-13](#_Toc464984167)

[Figure 3: This is the current pass transistor output stage 5-15](#_Toc464984168)

[Figure 9‑1: PCB design before printing 10-26](file:///C:\GIT\digital-power-supply\wiki\Edesign344Project.docx#_Toc464984169)

# 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.

# Bibliography

|  |  |
| --- | --- |
| [1] | “LM358,” [Online]. Available: www.ti.com/lit/ds/symlink/lm158-n.pdf. |
| [2] | D. Jones. [Online]. Available: https://youtu.be/tF2krfxYc68. |

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)