

Board 4 - Final

Nikolai Woeger

Plan of Record

The board was designed to perform the following functions:

- Power through usb-mini connector
- Power through barrel jack connector
- Power led
- Switch between power from barrel jack and usb
- Test power from barrel jack connector
- Test power from 2 terminal connector
- Switch between power test mediums
- Use an atmega328p as a programmable microcontroller
- Use a ch340g to communicate with the microcontroller - converting USB to UART
- Isolation switches to turn on/off the ch340g, microcontroller, and opamp for debugging
- A reset button for the arduino
- Switch to enable/disable the debouncing capacitor for the reset button
- An ICSP header to bootload the board
- Device under test circuit to calculate the thevenin resistance of a power supply
 - DAC to drive an analog voltage to the opamp (and therefore the MOSFET) determining the output current
 - Opamp to produce the current necessary for the MOSFET
 - MOSFET used to drive high currents from DUT power supply through the sense resistor
 - ADC to measure VRM voltage as well as voltage across the sense resistor
 - Able to measure the thevenin resistance at different currents
 - Able to send data back to the arduino using the ADC to be able to output the resistance measurements
- Female header pins for possible shield usage
- Rx and Tx LEDs to show UART communication line activity
- LED on D13 for blink code
- Test points on communication lines, crystal outputs, DUT voltage, and MOSFET voltage
- Buzzer and Smart LED outputs

Extra Functions:

- Gyroscope on I2C bus to track motion and report back to the arduino
- OLED display as an output for the user to understand board functionality
- Additional buttons to control the board state and perform certain functions

High level functionality:

- On Reset, each peripheral is initialized properly on the I2C bus (ADC, DAC, OLED Display, gyroscope)

- The gyroscope's motion vectors are polled by the arduino, and when the board is in motion, the LEDs flash, the buzzer beeps, and the screen says that the board is in motion
- When the board is not in motion, there is no beeping, LEDs are static, and the screen displays that the board is not in motion
- When a button is pressed, the board starts to test the DUT, sending data over the serial monitor to be plotted
 - When this process concludes, the average thevenin resistance is displayed on the screen

Important Components

Microcontroller: atmega328p

USB-UART bridge: ch340g

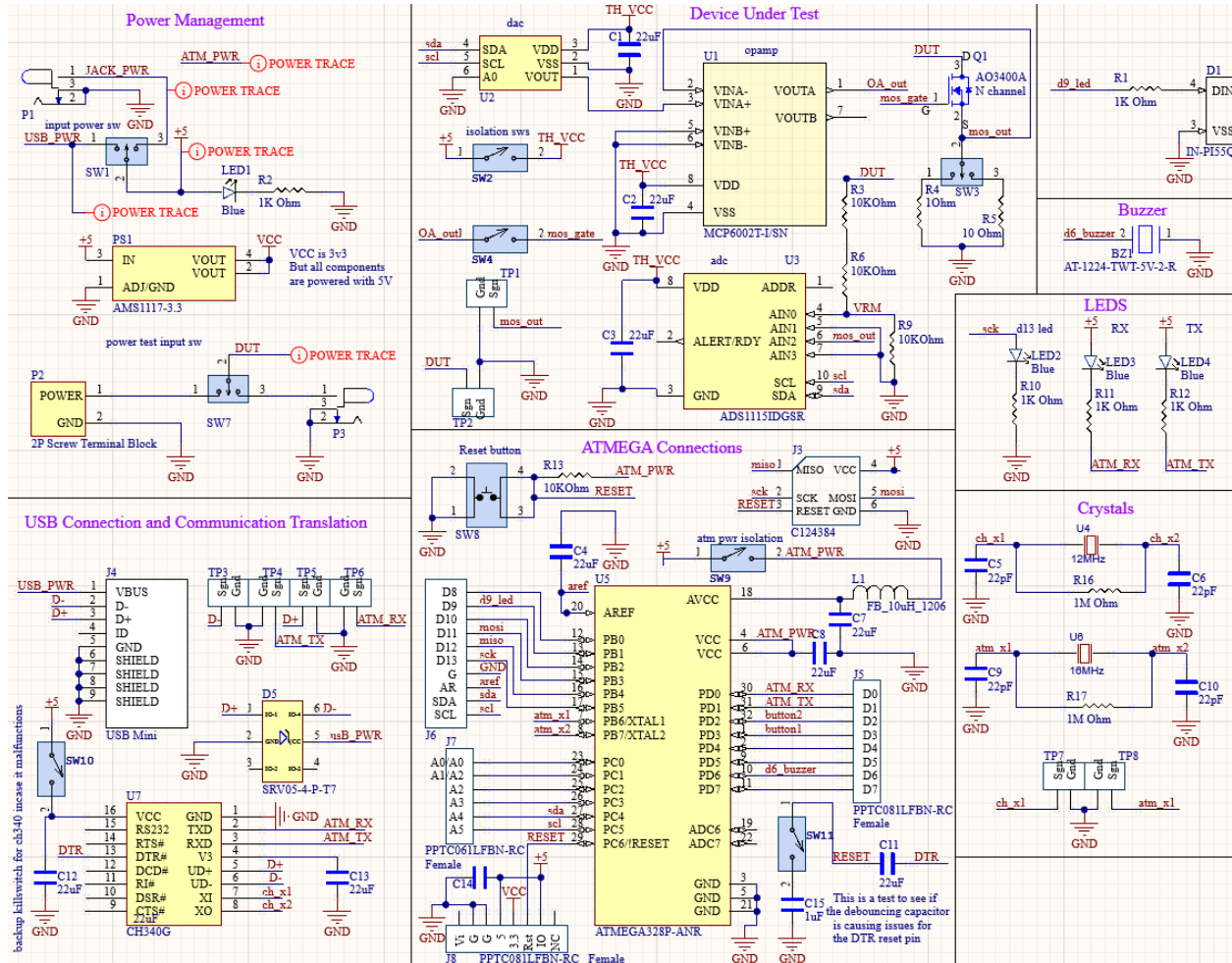
DAC: MCP4725

ADC: ADS1x15

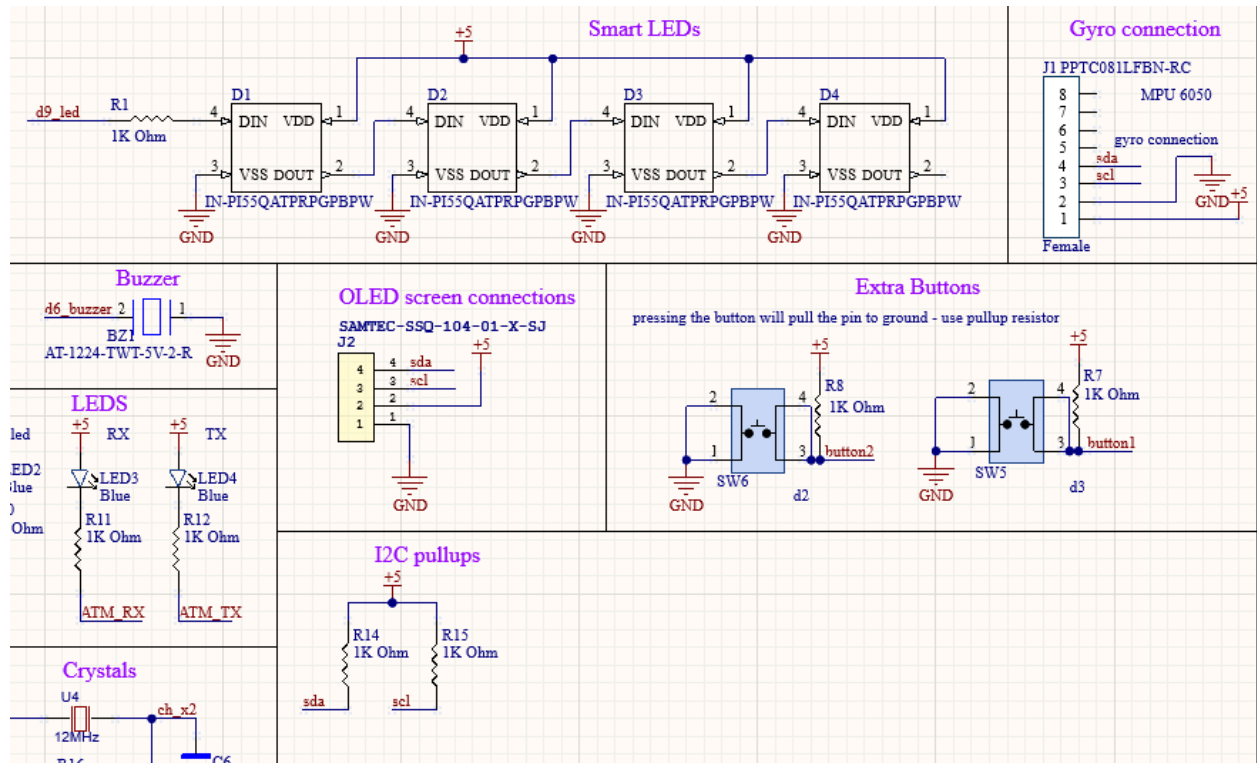
Gyroscope: MPU6050

OLED display: SSD1306

Schematic

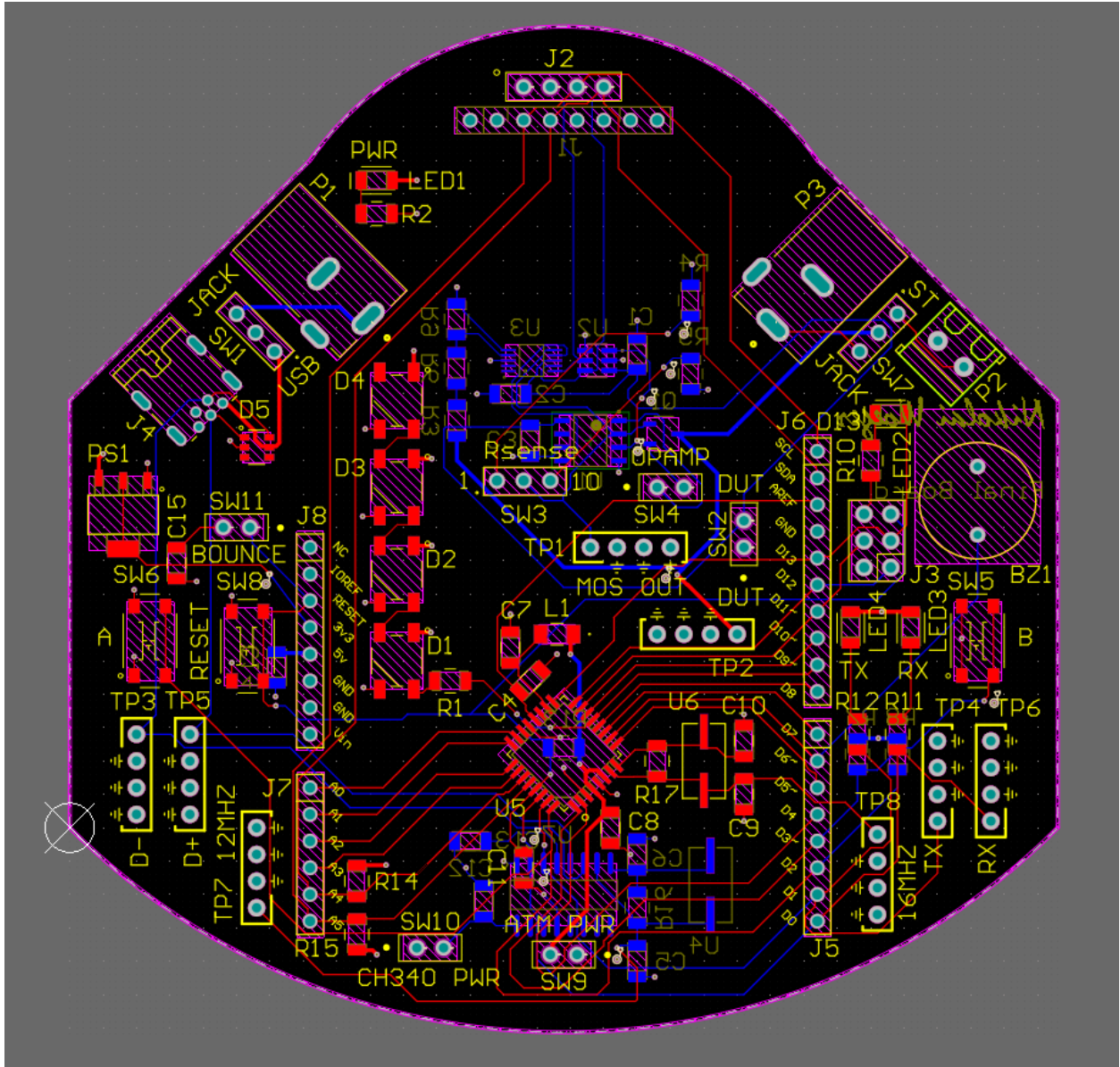


Above is the first part of the schematic, cut off for readability.



Above is the remaining schematic.

Layout

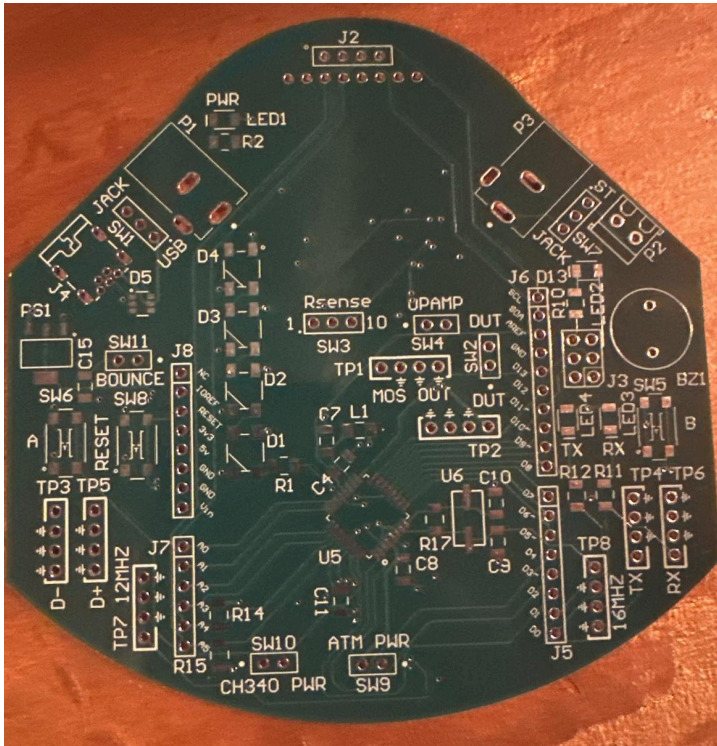


Here is the general layout of the board with just the top and bottom signal layers being shown (the middle layers essentially blind the view of everything else).

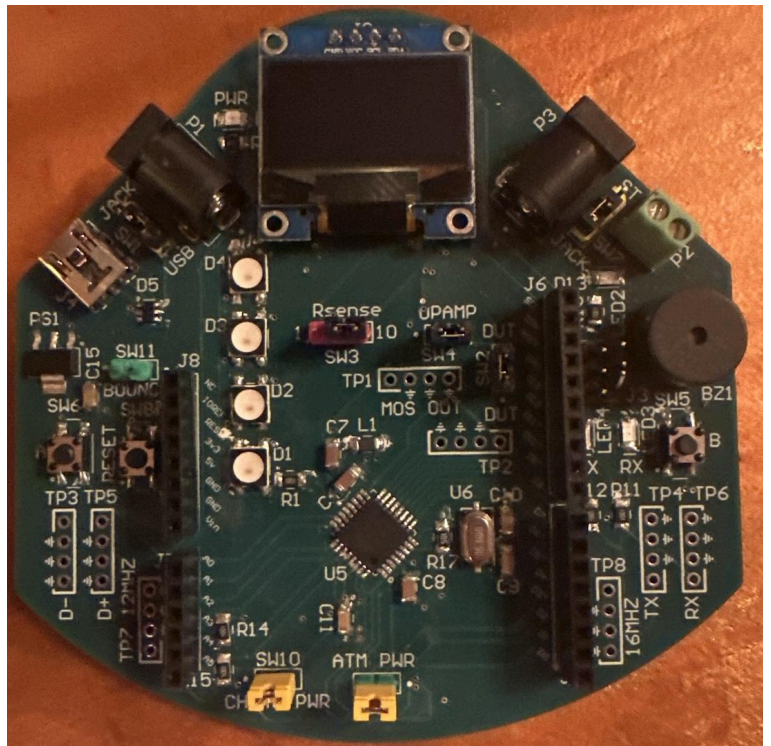
More layout photos are shown in the Appendix

Physical Board

Pre-Assembly

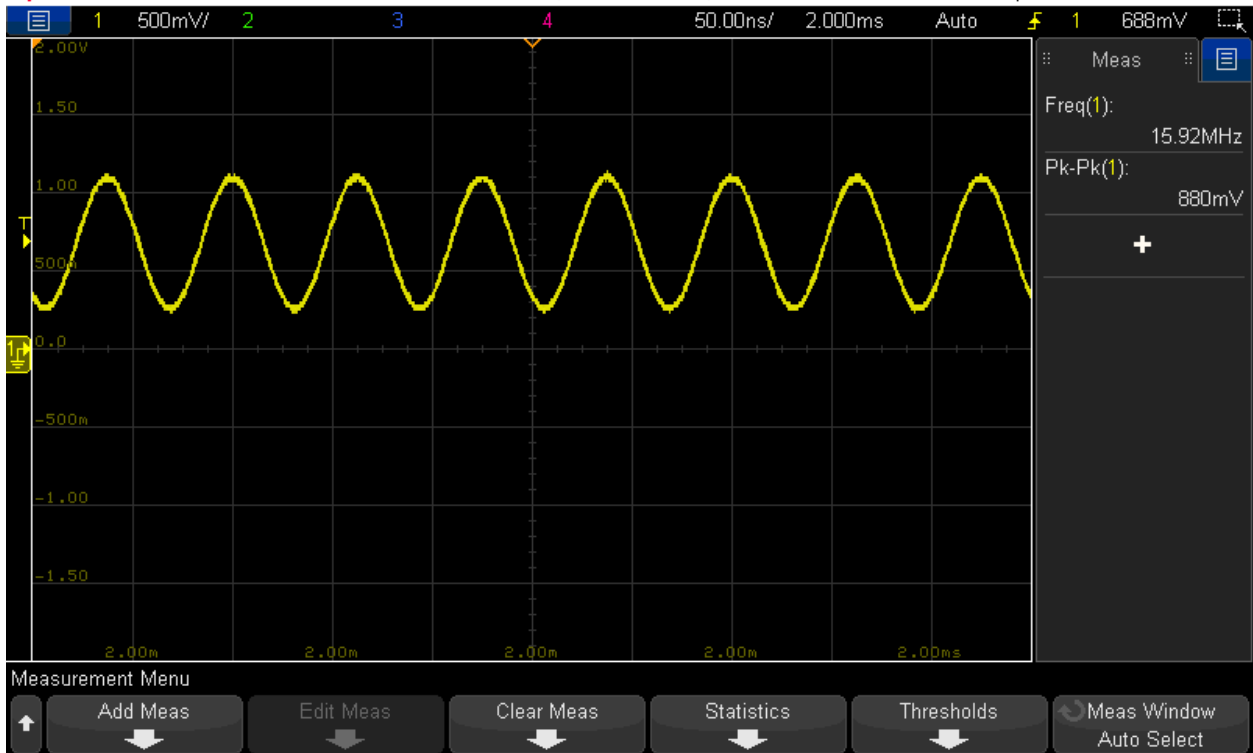


Post-Assembly

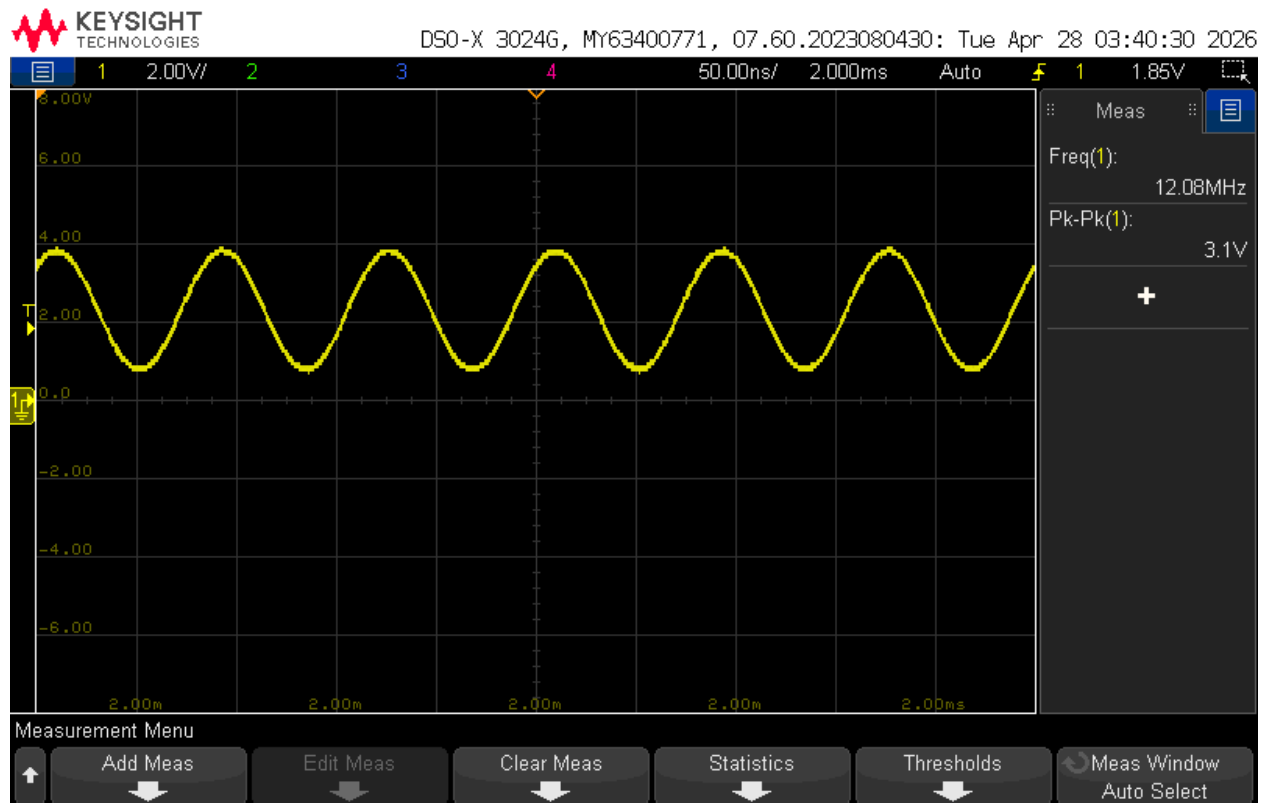


Initial Board Testing

When the board was first powered on and bootloaded, in order to ensure that both the ch340g and the atmega328 microcontroller were working properly, their external crystals were probed. Atmega 16MHz Crystal:



Ch340 12MHz Crystal:



With both of these crystals oscillating at the correct frequencies, the chips are confirmed to be functioning at a basic level and the board can now be programmed.

What I Learned

Before I show you exactly what I was able to accomplish with this board, I wanted to take a moment to take you through what I did in my design and assembly that went well, and what didn't. Specifically, I will take you through all of the major issues that I had with this board, and how I resolved them.

USB-UART Upload Issue

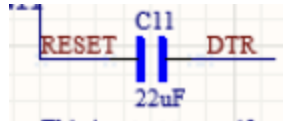
After bootloading the board, I immediately attempted to upload a simple “blink” program to test the LED on pin 13 and see if uploading would work. However, whenever I attempted an upload, I would get this error:

```
avrdude: stk500_disable(): unknown response=0x12  
Failed chip erase: uploading error: exit status 1
```

After doing some research, and attempting a “upload using programmer” upload, I knew that my atmega328p was working properly, as the blink code was executed. This meant that the communication from the USB was not working.

When this happened, I did what is called a “loop-back test”, which essentially just connects the RX and TX lines of the arduino together and sends data on the bus from the USB side. The consensus is that if you receive data back that is equal to the data you sent, the ch340g is also working properly. The problem is, this test succeeded. The problem was somewhere else.

After thinking more about how the ch340g works, I started to investigate the DTR pin. I probed the reset line whenever I attempted an upload, and discovered that it was never being pulled low, as I was expecting it too. After doing more research, I discovered that my capacitor in series on the DTR pin was likely too big.



While I had a 22uF capacitor, most websites using the ch340g recommend a 100nF capacitor. Knowing this, I desoldered the old capacitor and replaced it with a 47nF capacitor, the next readily available option.

At this point, I was able to upload to my board without a problem.

Code Freezing

After I had modularly created code for all of the separate peripherals of the board, I began putting all of the parts together and running them as a loop-implemented state machine. However, my initializations started to fail. My OLED display would print a “failed to initialize” message, and shortly after, the rest of the code would not execute properly. At first, I thought there might be address conflicts on the I2C bus, but this was quickly resolved as I confirmed they were different addresses. I attempted to allow the bus to settle before performing more initializations, but no amount of time solved the problem.

At this point, I went deeper into the SSD1306 library, the one that failed upon initialization. I discovered that the initialization struct created ~1kB in global variables due to its frame buffer. That was the problem. The Atmega328p is not meant for heavy memory usage, and only has 2kB of free memory to be used. I immediately optimized my code for memory, getting rid of unnecessary globals, using macros more often, putting strings in flash memory, etc. Once I made these memory optimizations, I significantly cut down on my global variable usage, and allowed room for local variables to exist and operate on the stack.

Putting strings in flash was really the trick for this, as I was printing many strings throughout the code’s execution.

Summary

These two issues were the main problems that I encountered during this project. What I learned from them was the following:

- To always pay attention to a microcontroller’s capabilities, including how communication protocols work, memory limitations, speed limitations, etc.

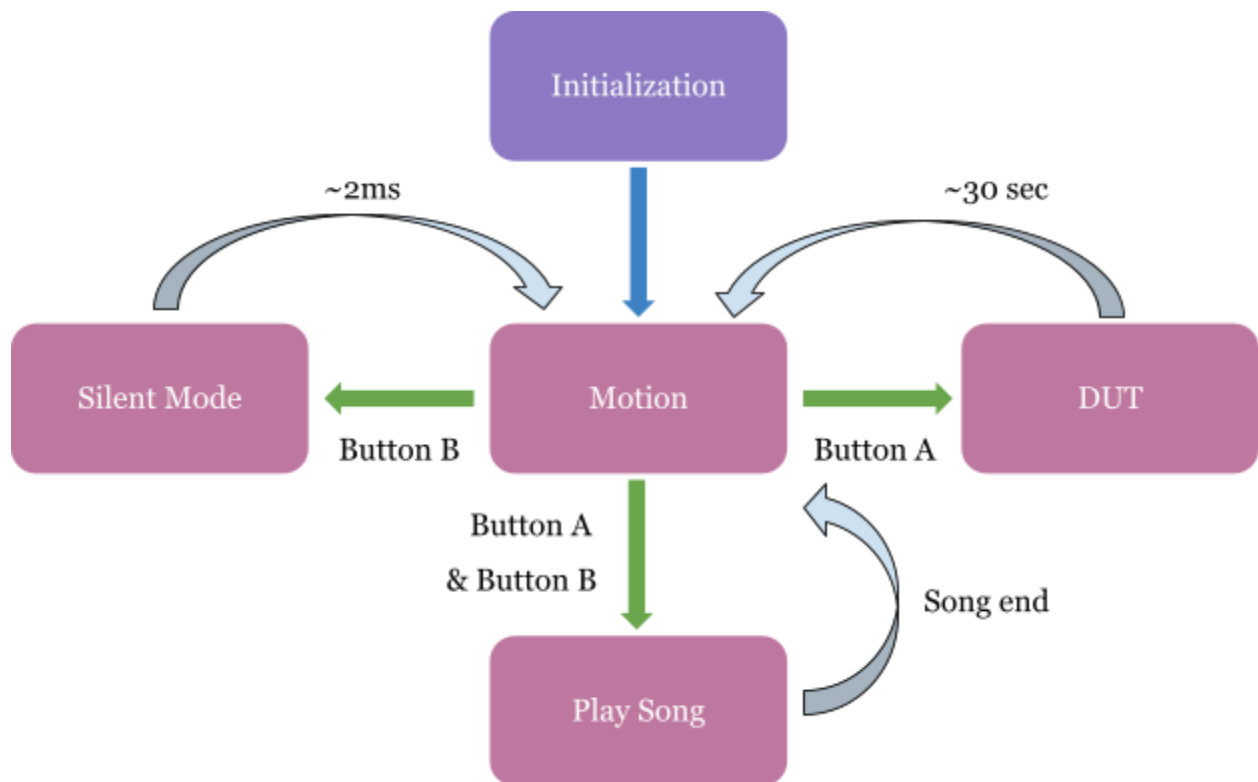
- I also learned that capacitor values MATTER in certain circumstances. Especially if they aren't necessarily decoupling.

In the design process, I very much enjoyed routing a 4-layer board, and I believe that is something that I will do again in the future to make designs better for signal integrity, as well as easy to follow. The 4-layer board truly allowed me to create the board that I wanted to, with enough space for all of my components without significant cross-unders.

High-Level Functionality

In order to meet the requirements set in the Plan of Record, a loop-implemented state machine was created to allow for users to perform specific board functions.

Here is a basic overview of the various states for the board, and how transitions between states happen:



With this overview of the various states, we will now explore what happens in each state.

Motion:

In this state, the gyroscope's acceleration vectors are constantly being polled. If the magnitude of the gyroscope is different than the baseline magnitude calculated during initialization, then the smart LEDs flash purple, the buzzer beeps (if not in silent mode), and the screen displays that motion is occurring. Otherwise, the LEDs remain static green, the buzzer remains off, and the screen displays that the device is still.

Silent Mode:

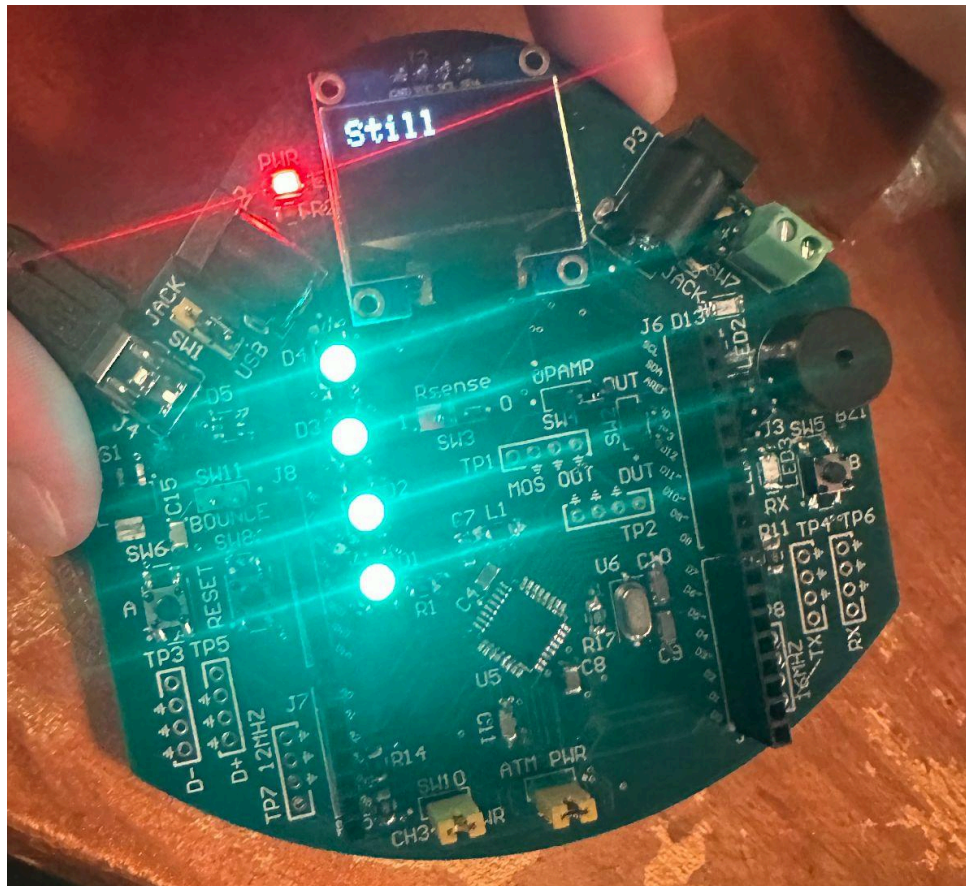
This state simply flips a boolean (silentmode) and then returns back to the motion state. This makes the buzzer not beep if motion is detected. Silent mode is achieved by pressing the B button when the device is in the motion state.

DUT:

This state stops the motion polling and starts testing the power supply. It returns relevant values to the serial monitor for plotting. When it concludes, the right half of the screen is updated displaying the average thevenin resistance of the power supply. DUT mode is achieved by pressing button A when in the motion state.

Play Song:

This mode uses a buzzer to play “Für Elise” from a converted midi file. This mode is achieved by holding button A until you press button B, then releasing button A, then releasing button B. (Essentially a dumb way of detecting if they are both pressed). Once the song is finished, the state returns to the motion loop.

Motion:

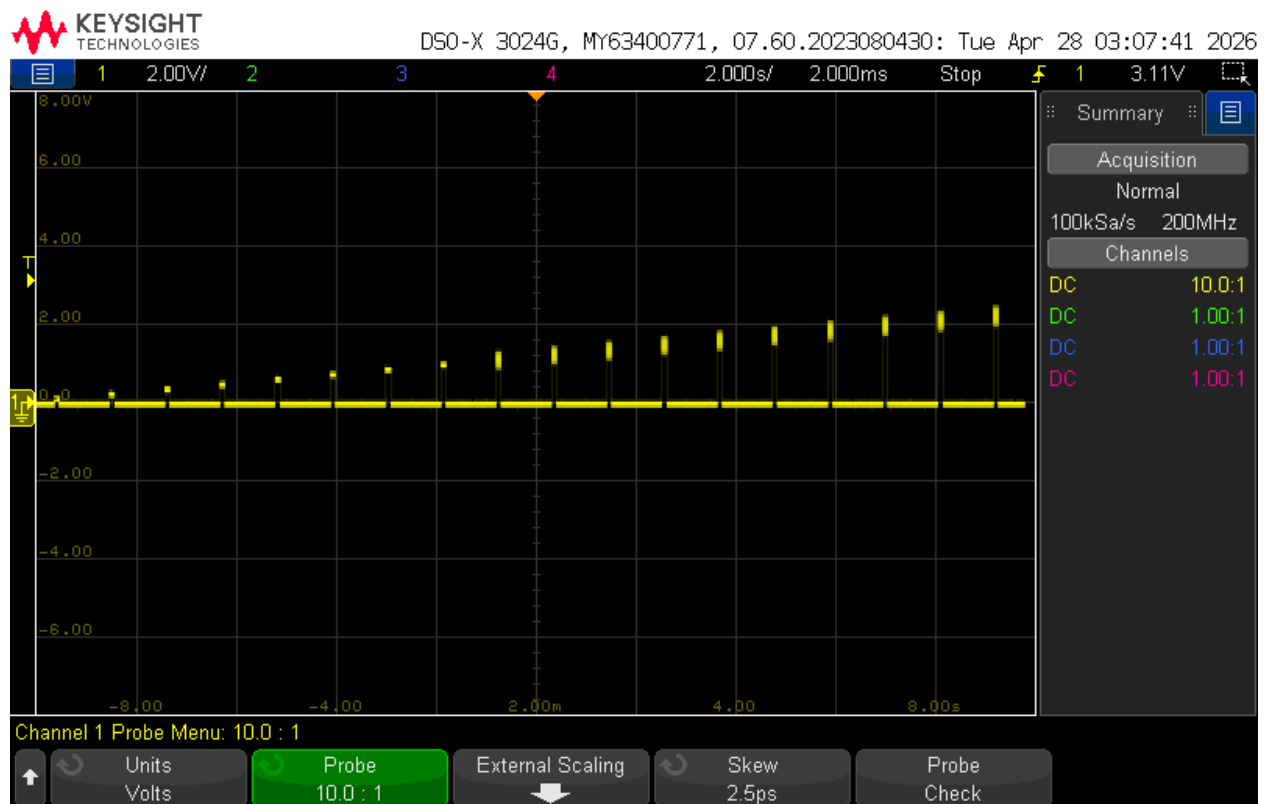
Device Under Test Functionality

This portion of the project turns the mosfet on for a short duration with varying gate voltages, then measures the current through the sense resistor. Then, it uses this measurement in conjunction with the measurement of the thevenin voltage in order to perform a calculation to obtain the thevenin resistance.

As the current is ramped up, the thevenin resistance may change, so these values are sent back through Serial, and can be plotted on the Arduino IDE's serial plotter.

To ensure that the DUT circuit is working, we cause the device to begin turning on the MOSFET at varying voltages, and then measure the MOSFET output. If we get a square wave that ramps up slowly, then we know that the following is functioning correctly:

- Microcontroller (to send commands)
- DAC output
- I2C bus
- The Op-Amp
- The MOSFET



Seeing this ramping square wave asserts that our DUT circuit is at least mostly functional!

The only thing left to check is the ADC measurements, which we can use a serial plotter. We are expecting very small resistances for this 5V power supply. The thevenin voltage and the load voltage should both be close to 5V for all of the data points as well.

When the “A” button is pressed, then the DUT test code begins to run, causing the above output at the source of the MOSFET. In the serial monitor, the following is printed:

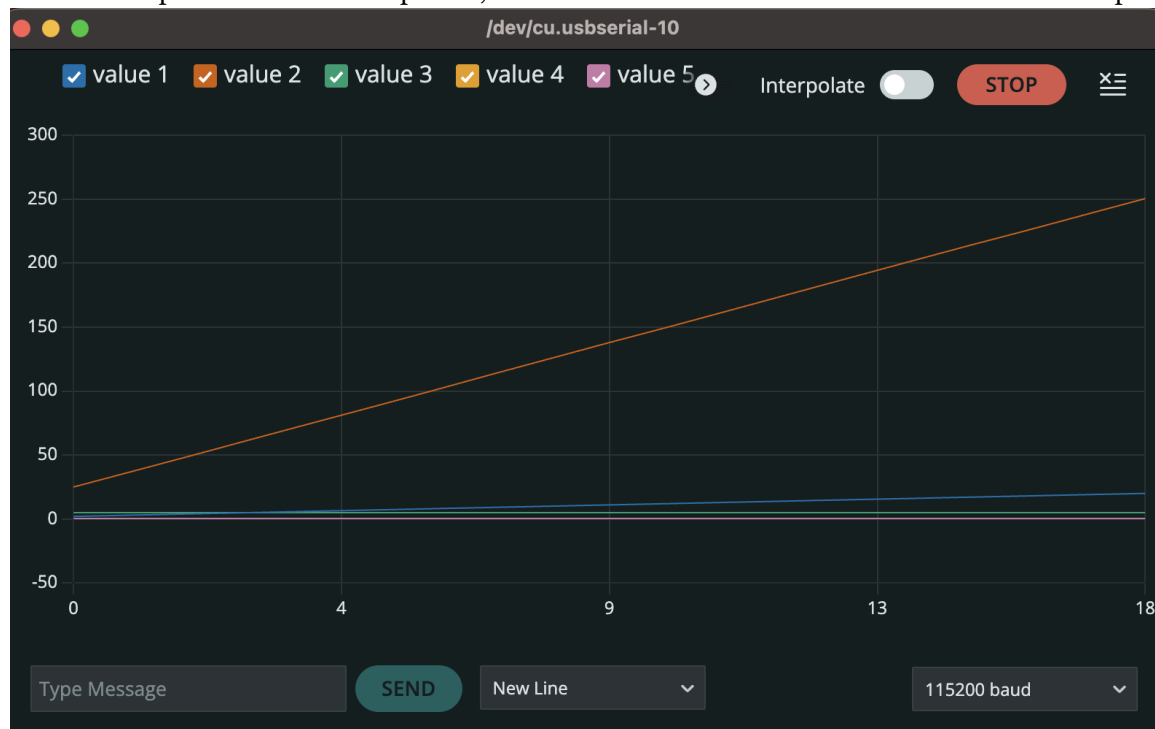
```
1, 12.772, 4.9952, 4.9904, 0.3710
2, 25.070, 4.9952, 4.9860, 0.3654
3, 37.484, 4.9952, 4.9814, 0.3678
4, 49.918, 4.9952, 4.9769, 0.3663
5, 62.574, 4.9952, 4.9723, 0.3661
6, 75.154, 4.9952, 4.9676, 0.3666
7, 87.632, 4.9952, 4.9631, 0.3653
8, 100.195, 4.9952, 4.9586, 0.3653
9, 113.120, 4.9952, 4.9541, 0.3638
10, 125.726, 4.9952, 4.9490, 0.3674
11, 138.303, 4.9952, 4.9442, 0.3688
12, 150.732, 4.9952, 4.9450, 0.3327
13, 163.419, 4.9952, 4.9349, 0.3689
14, 175.947, 4.9952, 4.9310, 0.3652
15, 188.733, 4.9952, 4.9261, 0.3661
16, 201.343, 4.9952, 4.9217, 0.3651
17, 213.821, 4.9952, 4.9167, 0.3669
18, 226.203, 4.9952, 4.9122, 0.3670
19, 238.761, 4.9952, 4.9073, 0.3683
20, 251.242, 4.9952, 4.9031, 0.3667
```

This data itself does not tell us much, but the values follow the following scheme per line:

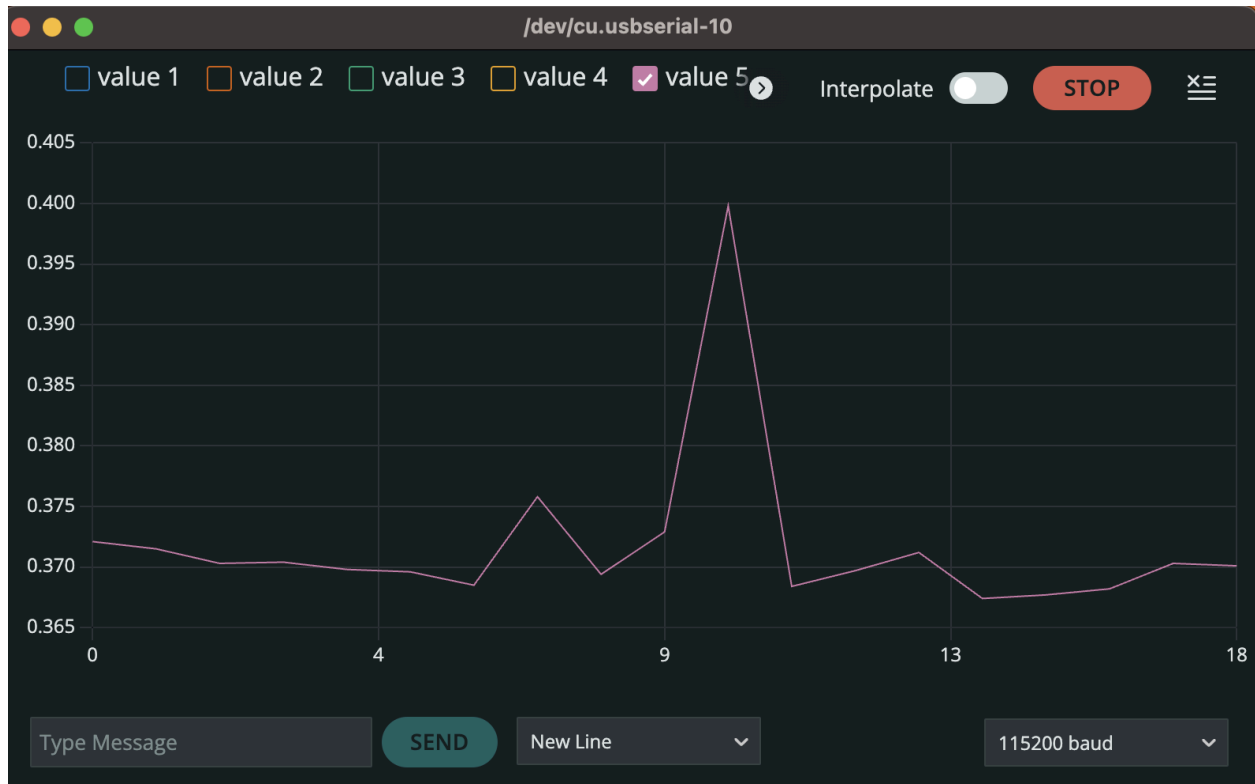
Trial #, load current, thevenin voltage, load voltage, thevenin resistance

With this in mind, the values are matching expectations

When this is plotted on the serial plotter, then the resistance can be tracked as the current ramps up:



We still cannot make sense of this, so if the resistance is the only value we see, then it looks like this:

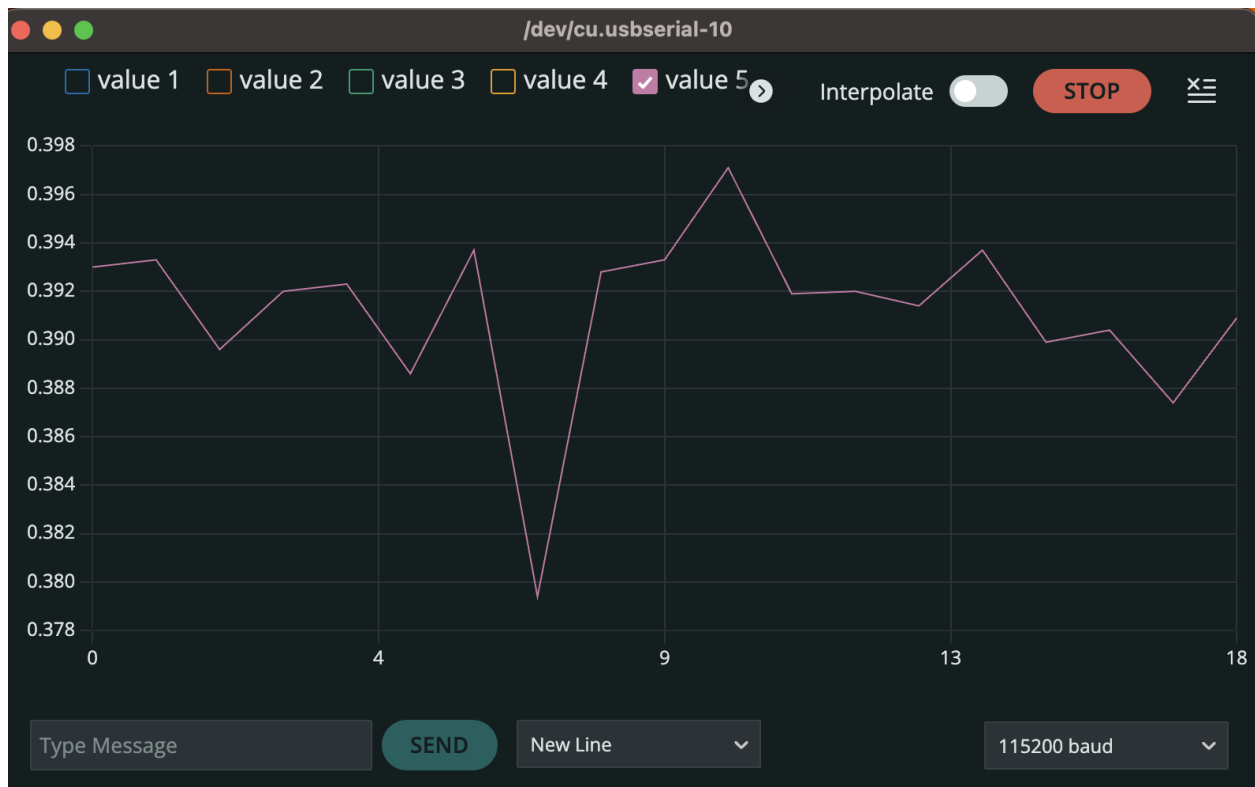


As you can see, the thevenin resistance of this power supply stays relatively consistent, with a spike of around 0.03 Ohms at around 150 mA of current. This is a good power supply!

Now, more tests can be done to determine how good other power supplies are:

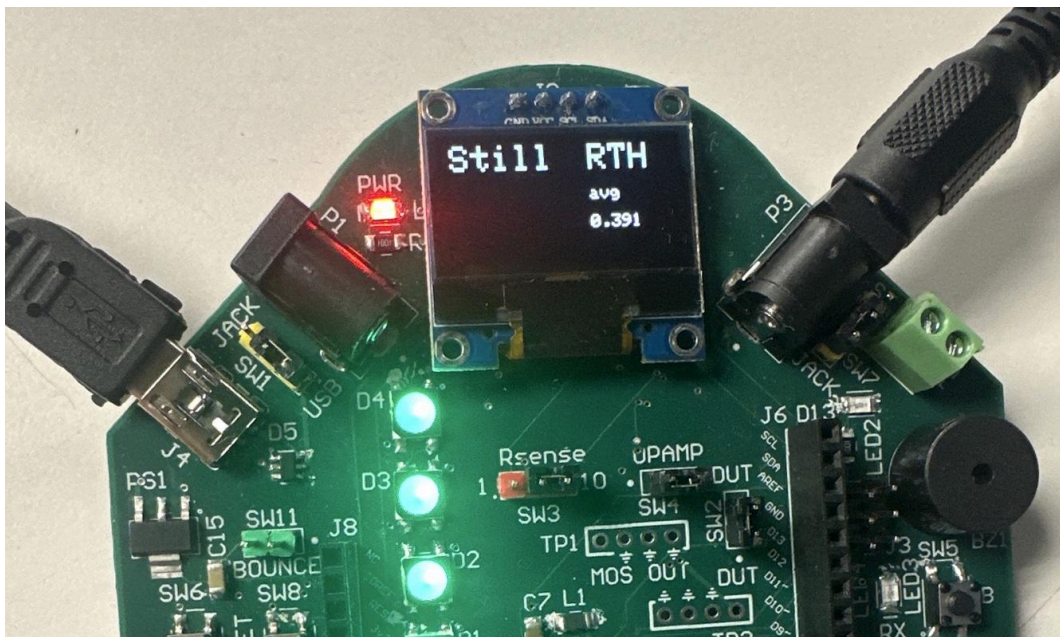


Above is a battery producing about 8V, also a pretty consistent supply.



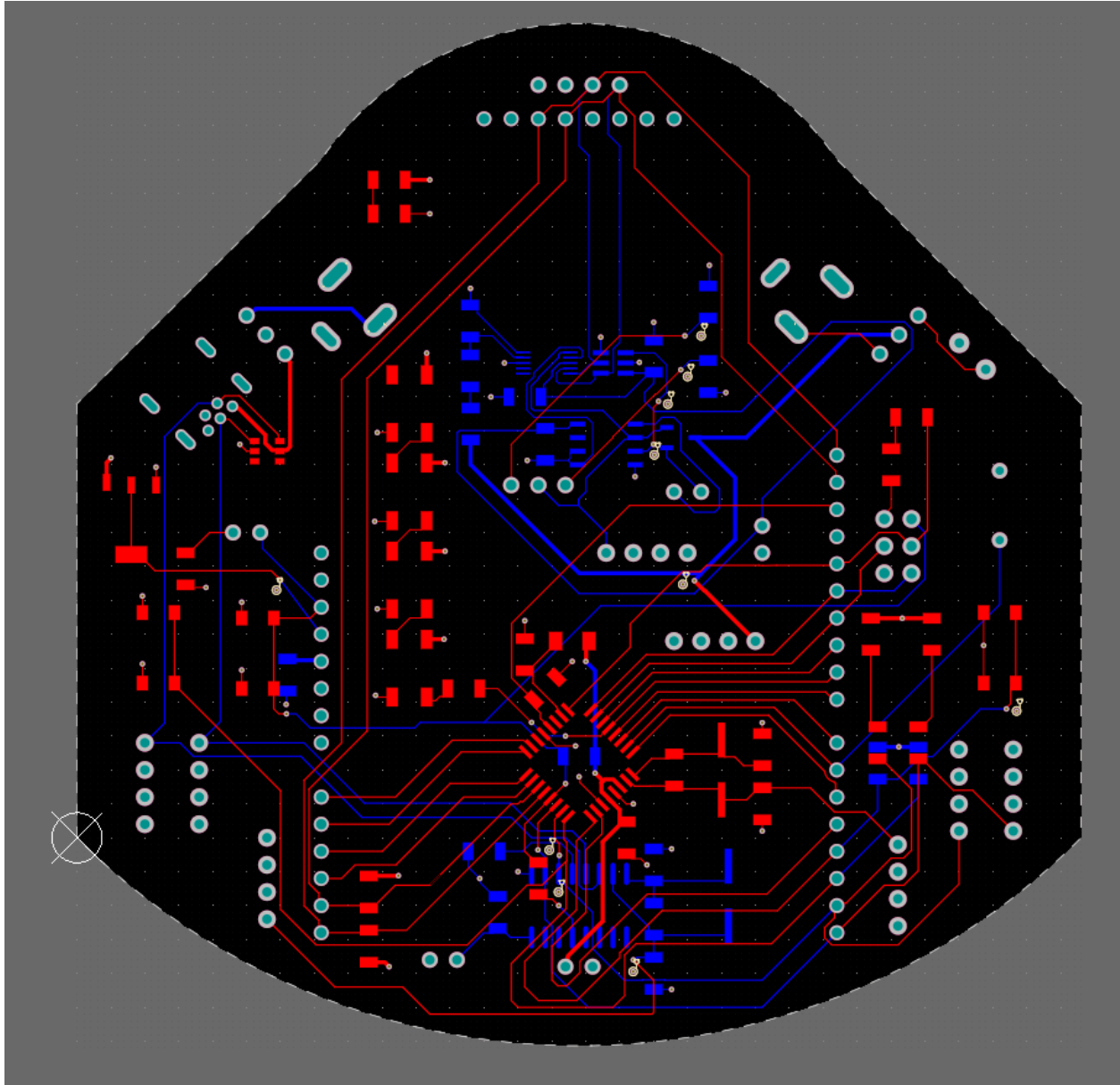
Here is a 12V power supply, showing some noise but still relatively consistent.

In addition to being able to plot to power supply, this board also takes an average of the thevenin resistance for each power supply that it tests, it then displays that number onto the OLED display:

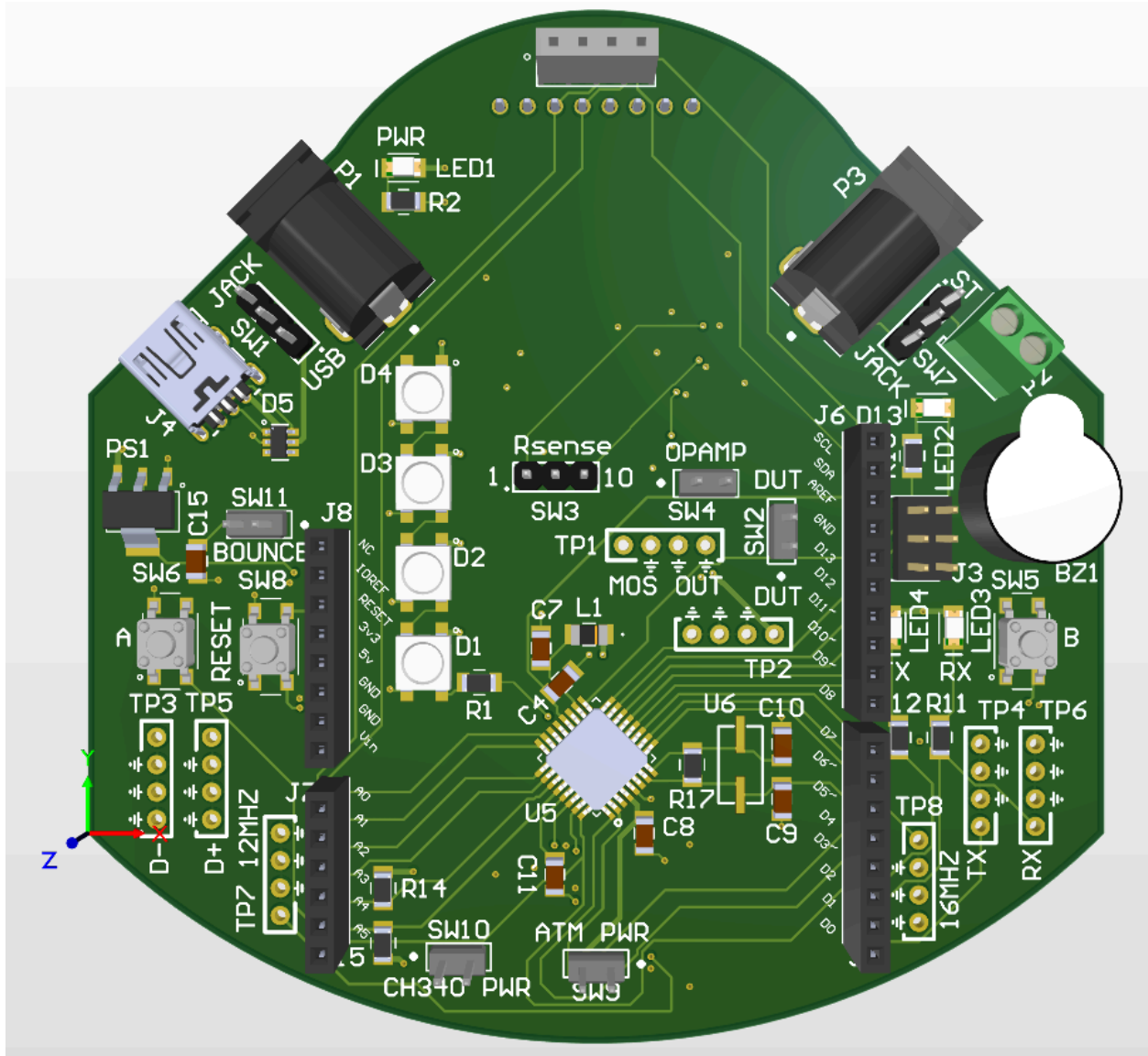


Appendix

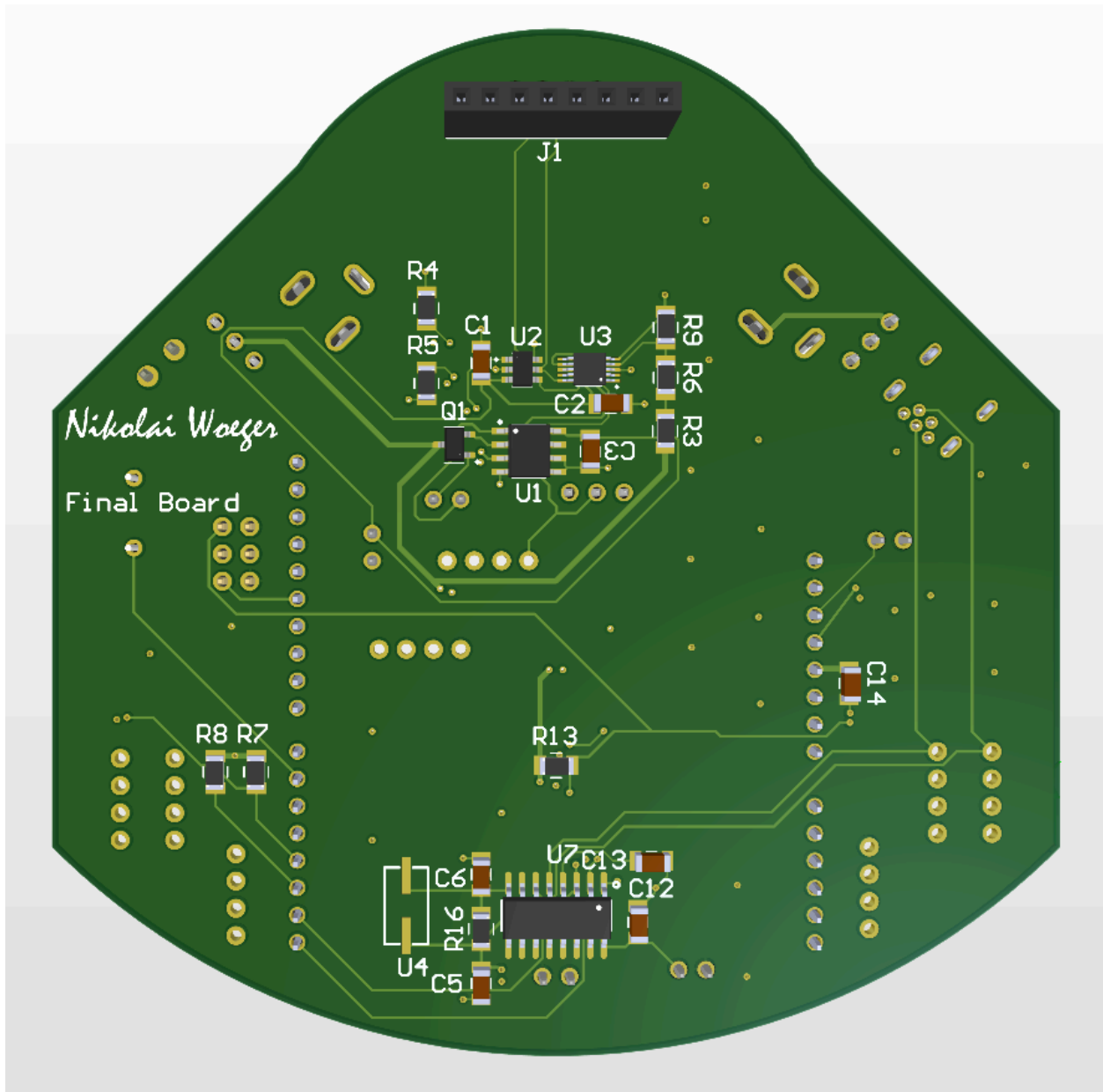
Layout Images



Layout: Signal Layers



Top 3D



Bottom 3D

Code

The code is broken up into 5 files:

Main.ino, dut.ino, gyro.ino, oled.ino, io.ino

Main.ino:

```
#include <Wire.h>
#include <Adafruit_ADS1X15.h>
#include <avr/pgmspace.h>
```

```
#define BUZZER 6
#define BUTTONA 2
#define BUTTONB 3

Adafruit_ADS1115 ads;
bool moving = false;

struct PollReturn{
    bool moving;
    float magnitude;
};

#define ARRAY_LEN(array) (sizeof(array) / sizeof(array[0]))
#define E5 659
#define Db5 622
#define B4 494
#define D5 587
#define C5 523
#define A4 440
#define C4 262
#define E4 330
#define Gb4 415
#define G4 392
#define F5 698
#define F4 349
#define E6 1319
#define D4 294
#define Ab5 932
#define A5 880
#define G5 784
#define Ab4 466
#define C6 1047
#define B5 988
#define Fb4 370
#define Cb5 554
#define Db4 311
#define A3 220
#define D6 1175
#define A6 1760
#define C7 2093
#define E7 2637
```

```
#define D7 2349
#define B6 1976
#define Ab6 1865
#define Gb6 1661
#define G6 1568
#define Fb6 1480
#define F6 1397
#define Db6 1245
#define Cb6 1109
#define Gb5 831
#define Fb5 740

// song stored in flash memory (converted from a midi file)
// sorry for its length :)
const int midi1[611][3] PROGMEM= {
{E5, 192, 13},
{Db5, 196, 0},
{E5, 192, 13},
{Db5, 196, 4},
{E5, 196, 0},
{B4, 192, 12},
{D5, 196, 4},
{C5, 196, 4},
{A4, 396, 204},
{C4, 196, 4},
{E4, 196, 0},
{A4, 192, 8},
{B4, 392, 208},
{E4, 192, 8},
{Gb4, 192, 13},
{B4, 196, 0},
{C5, 392, 208},
{E4, 192, 8},
{E5, 192, 13},
{Db5, 196, 4},
{E5, 196, 4},
{Db5, 196, 4},
{E5, 196, 0},
{B4, 192, 13},
{D5, 196, 0},
{C5, 192, 8},
{A4, 392, 213},
```

```
{C4, 196, 4},
{E4, 196, 4},
{A4, 196, 0},
{B4, 592, 13},
{C5, 196, 4},
{B4, 196, 0},
{C4, 392, 413},
{E5, 199, 4},
{Db5, 210, 5},
{E5, 213, 5},
{Db5, 213, 0},
{E5, 208, 9},
{B4, 196, 8},
{D5, 192, 12},
{C5, 196, 0},
{A4, 392, 213},
{C4, 196, 0},
{E4, 192, 8},
{A4, 192, 12},
{B4, 396, 204},
{E4, 196, 4},
{Gb4, 196, 4},
{B4, 196, 4},
{C5, 396, 204},
{E4, 196, 0},
{E5, 192, 8},
{Db5, 192, 12},
{E5, 196, 4},
{Db5, 196, 0},
{E5, 192, 13},
{B4, 196, 0},
{D5, 192, 12},
{C5, 196, 4},
{A4, 396, 200},
{C4, 192, 8},
{E4, 192, 8},
{A4, 192, 12},
{B4, 396, 204},
{C5, 192, 8},
{B4, 192, 8},
{C4, 396, 204},
{B4, 196, 0},
```

```
{C5, 192, 13},
{D5, 196, 0},
{E5, 592, 8},
{G4, 192, 8},
{F5, 192, 8},
{E5, 192, 12},
{D5, 596, 4},
{F4, 196, 4},
{E5, 196, 4},
{D5, 196, 0},
{C5, 592, 8},
{E4, 192, 8},
{D5, 192, 12},
{C5, 196, 0},
{B4, 192, 413},
{E4, 196, 4},
{E5, 196, 404},
{E5, 196, 4},
{E6, 196, 400},
{Db5, 192, 12},
{E5, 196, 400},
{Db5, 192, 12},
{E5, 197, 4},
{Db5, 202, 0},
{E5, 200, 9},
{Db5, 205, 14},
{E5, 226, 5},
{B4, 229, 5},
{D5, 229, 0},
{C5, 225, 10},
{A4, 427, 217},
{C4, 194, 8},
{E4, 192, 8},
{A4, 192, 12},
{B4, 396, 204},
{E4, 196, 0},
{Gb4, 192, 12},
{B4, 196, 4},
{C5, 396, 204},
{E4, 196, 0},
{E5, 192, 12},
{Db5, 196, 4},
```

```
{E5, 196, 0},
{Db5, 192, 13},
{E5, 196, 0},
{B4, 192, 13},
{D5, 196, 4},
{C5, 196, 4},
{A4, 396, 200},
{C4, 192, 8},
{E4, 192, 12},
{A4, 196, 0},
{B4, 392, 208},
{C5, 192, 13},
{B4, 196, 4},
{C4, 396, 200},
{B4, 192, 8},
{C5, 192, 8},
{D5, 192, 8},
{E5, 592, 8},
{G4, 192, 8},
{F5, 192, 13},
{E5, 196, 0},
{D5, 592, 12},
{F4, 196, 4},
{E5, 196, 4},
{D5, 196, 0},
{C5, 592, 13},
{E4, 196, 0},
{D5, 192, 8},
{C5, 192, 12},
{B4, 296, 304},
{E4, 96, 100},
{E5, 192, 412},
{E5, 96, 104},
{E6, 96, 500},
{Db5, 92, 108},
{E5, 192, 408},
{Db5, 93, 117},
{E5, 204, 0},
{Db5, 97, 122},
{E5, 214, 0},
{Db5, 217, 14},
{E5, 229, 0},
```

```
{B4, 234, 10},
{D5, 236, 15},
{C5, 118, 121},
{A4, 340, 340},
{C4, 201, 13},
{E4, 198, 0},
{A4, 192, 8},
{B4, 392, 208},
{E4, 192, 8},
{Gb4, 192, 13},
{B4, 196, 4},
{C5, 396, 204},
{E4, 96, 104},
{E5, 196, 4},
{Db5, 196, 0},
{E5, 192, 8},
{Db5, 192, 12},
{E5, 196, 4},
{B4, 196, 4},
{D5, 196, 0},
{C5, 192, 8},
{A4, 292, 313},
{C4, 196, 0},
{E4, 192, 8},
{A4, 192, 8},
{B4, 392, 212},
{D4, 196, 4},
{C5, 196, 4},
{B4, 96, 100},
{C4, 296, 304},
{E4, 96, 101},
{C5, 98, 104},
{G4, 100, 106},
{C5, 797, 136},
{F5, 358, 5},
{E5, 116, 5},
{E5, 467, 0},
{D5, 329, 118},
{Ab5, 317, 14},
{A5, 104, 5},
{A5, 213, 0},
{G5, 208, 9},
```

```
{F5, 208, 9},
{E5, 208, 14},
{D5, 213, 0},
{C5, 208, 9},
{Ab4, 426, 14},
{A4, 322, 5},
{Ab4, 104, 5},
{A4, 104, 5},
{G4, 104, 0},
{A4, 100, 14},
{Ab4, 104, 0},
{C5, 861, 14},
{D5, 213, 5},
{Db5, 213, 5},
{E5, 539, 113},
{E5, 213, 0},
{F5, 208, 9},
{A4, 100, 118},
{C5, 317, 122},
{D5, 104, 5},
{C5, 104, 5},
{B4, 104, 0},
{C5, 100, 9},
{D5, 317, 9},
{B4, 100, 14},
{C5, 104, 0},
{G5, 100, 14},
{G4, 104, 5},
{G5, 104, 5},
{A4, 104, 0},
{G5, 100, 9},
{B4, 100, 14},
{G5, 104, 0},
{C5, 100, 14},
{G5, 104, 0},
{D5, 100, 9},
{G5, 100, 9},
{E5, 100, 14},
{G5, 104, 0},
{C6, 100, 14},
{B5, 104, 5},
{A5, 104, 0},
```

```
{G5, 100, 9},
{F5, 100, 14},
{E5, 104, 5},
{D5, 104, 0},
{G5, 100, 9},
{F5, 100, 14},
{D5, 104, 5},
{C5, 104, 5},
{G5, 104, 5},
{G4, 104, 5},
{G5, 104, 0},
{A4, 100, 14},
{G5, 104, 5},
{B4, 104, 0},
{G5, 100, 9},
{C5, 100, 9},
{G5, 100, 9},
{D5, 100, 14},
{G5, 104, 5},
{E5, 104, 5},
{G5, 104, 5},
{C6, 104, 5},
{B5, 104, 0},
{A5, 100, 14},
{G5, 104, 0},
{F5, 100, 9},
{E5, 100, 9},
{D5, 100, 14},
{G5, 104, 0},
{F5, 100, 14},
{D5, 104, 0},
{E5, 100, 14},
{F5, 104, 0},
{E5, 100, 9},
{Db5, 100, 14},
{E5, 104, 5},
{B4, 104, 5},
{E5, 104, 5},
{Db5, 104, 5},
{E5, 104, 0},
{B4, 100, 14},
{E5, 104, 5},
```

```
{Db5, 104, 0},
{E5, 697, 11},
{B4, 253, 17},
{E5, 268, 6},
{Db5, 277, 0},
{E5, 882, 13},
{B4, 299, 13},
{E5, 312, 673},
{Db5, 319, 0},
{E5, 311, 606},
{Db5, 256, 16},
{E5, 239, 0},
{Db5, 209, 8},
{E5, 190, 8},
{B4, 189, 8},
{D5, 189, 12},
{C5, 193, 0},
{A4, 387, 210},
{C4, 193, 4},
{E4, 193, 4},
{A4, 193, 4},
{B4, 391, 197},
{E4, 189, 12},
{Gb4, 193, 4},
{B4, 193, 0},
{C5, 387, 206},
{E4, 189, 12},
{E5, 193, 4},
{Db5, 193, 0},
{E5, 189, 12},
{Db5, 193, 0},
{E5, 189, 12},
{B4, 193, 4},
{D5, 193, 4},
{C5, 193, 4},
{A4, 391, 197},
{C4, 189, 8},
{E4, 189, 8},
{A4, 189, 8},
{B4, 387, 206},
{D4, 189, 12},
{C5, 193, 4},
```

```
{B4, 193, 4},
{C4, 391, 197},
{B4, 189, 12},
{C5, 193, 4},
{D5, 193, 4},
{E5, 588, 4},
{G4, 193, 4},
{F5, 193, 0},
{E5, 189, 12},
{D5, 588, 0},
{F4, 189, 8},
{E5, 189, 12},
{D5, 193, 4},
{C5, 588, 0},
{E4, 189, 12},
{D5, 193, 0},
{C5, 189, 8},
{B4, 189, 407},
{E4, 193, 0},
{E5, 189, 407},
{E5, 193, 0},
{E6, 189, 403},
{Db5, 189, 12},
{E5, 193, 399},
{Db5, 193, 4},
{E5, 193, 0},
{Db5, 189, 8},
{E5, 189, 8},
{Db5, 189, 12},
{E5, 193, 0},
{B4, 189, 8},
{D5, 189, 8},
{C5, 189, 12},
{A4, 588, 0},
{C4, 189, 8},
{E4, 189, 8},
{A4, 189, 12},
{B4, 391, 197},
{E4, 189, 12},
{Gb4, 193, 4},
{B4, 193, 4},
{C5, 391, 201},
```

```
{E4, 193, 4},
{E5, 193, 0},
{Db5, 189, 12},
{E5, 193, 4},
{Db5, 193, 0},
{E5, 189, 12},
{B4, 193, 4},
{D5, 193, 4},
{C5, 193, 4},
{A4, 588, 0},
{C4, 189, 8},
{E4, 189, 12},
{A4, 193, 0},
{B4, 387, 210},
{C5, 189, 8},
{B4, 189, 8},
{C4, 391, 789},
{G4, 983, 197},
{A4, 588, 197},
{E5, 193, 0},
{D5, 193, 0},
{F5, 588, 197},
{F5, 193, 197},
{A4, 983, 197},
{F4, 588, 201},
{E4, 193, 4},
{D4, 193, 0},
{Fb4, 584, 206},
{C4, 193, 197},
{A4, 193, 197},
{E4, 193, 197},
{B4, 193, 197},
{A4, 781, 403},
{Ab4, 983, 197},
{F4, 588, 197},
{Cb5, 193, 0},
{D5, 193, 0},
{D5, 588, 201},
{D5, 193, 197},
{D5, 983, 197},
{Db5, 588, 197},
{D5, 193, 0},
```

```
{Db4, 189, 8},
{D4, 785, 0},
{D4, 391, 4},
{D4, 588, 197},
{Gb4, 193, 197},
{E4, 588, 592},
{B4, 387, 798},
{A3, 123, 12},
{C4, 127, 4},
{E4, 127, 4},
{A4, 127, 4},
{C5, 127, 0},
{E5, 123, 12},
{D5, 130, 0},
{C5, 129, 13},
{B4, 137, 5},
{A4, 141, 5},
{C5, 146, 5},
{E5, 151, 5},
{A5, 156, 0},
{C6, 156, 10},
{E6, 159, 11},
{D6, 166, 11},
{C6, 170, 11},
{B5, 170, 17},
{A5, 176, 0},
{C6, 172, 17},
{E6, 180, 0},
{A6, 177, 18},
{C7, 183, 6},
{E7, 183, 6},
{D7, 183, 6},
{C7, 183, 6},
{B6, 183, 6},
{Ab6, 183, 6},
{A6, 183, 6},
{Gb6, 182, 6},
{G6, 178, 6},
{Fb6, 176, 6},
{F6, 175, 0},
{E6, 167, 16},
{Db6, 167, 0},
```

```
{D6, 159, 11},
{Cb6, 156, 16},
{C6, 161, 5},
{B5, 161, 0},
{Ab5, 156, 16},
{A5, 161, 0},
{Gb5, 156, 10},
{G5, 153, 15},
{Fb5, 155, 5},
{F5, 151, 0},
{E5, 220, 9},
{Db5, 213, 14},
{E5, 211, 0},
{B4, 199, 13},
{D5, 197, 0},
{C5, 192, 13},
{A4, 396, 204},
{C4, 196, 4},
{E4, 196, 0},
{A4, 192, 8},
{B4, 392, 212},
{E4, 196, 4},
{Gb4, 196, 0},
{B4, 192, 13},
{C5, 396, 200},
{E4, 192, 8},
{E5, 192, 13},
{Db5, 196, 0},
{E5, 192, 8},
{Db5, 192, 13},
{E5, 196, 0},
{B4, 192, 8},
{D5, 192, 8},
{C5, 192, 8},
{A4, 392, 208},
{C4, 192, 8},
{E4, 192, 8},
{A4, 192, 13},
{B4, 396, 204},
{D4, 196, 4},
{C5, 196, 4},
{B4, 196, 0},
```

```
{A4, 396, 200},
{B4, 192, 13},
{C5, 196, 0},
{D5, 192, 8},
{E5, 592, 13},
{G4, 196, 4},
{F5, 196, 4},
{E5, 196, 0},
{D5, 592, 12},
{F4, 196, 4},
{E5, 196, 4},
{D5, 196, 4},
{C5, 596, 4},
{E4, 196, 0},
{D5, 192, 8},
{C5, 192, 8},
{B4, 192, 408},
{E4, 192, 8},
{E5, 192, 412},
{E5, 196, 4},
{E6, 196, 400},
{Db5, 192, 8},
{E5, 192, 408},
{Db5, 192, 12},
{E5, 196, 4},
{Db5, 196, 0},
{E5, 192, 8},
{Db5, 192, 8},
{E5, 192, 8},
{B4, 192, 8},
{D5, 192, 13},
{C5, 196, 4},
{A4, 596, 4},
{C4, 196, 0},
{E4, 192, 8},
{A4, 192, 8},
{B4, 592, 13},
{E4, 196, 0},
{Gb4, 192, 13},
{B4, 196, 0},
{C5, 392, 213},
{E4, 196, 0},
```

```

{E5, 192, 8},
{Db5, 192, 8},
{E5, 192, 13},
{Db5, 196, 0},
{E5, 192, 8},
{B4, 192, 12},
{D5, 196, 0},
{C5, 192, 12},
{A4, 405, 208},
{C4, 200, 13},
{E4, 207, 4},
{A4, 211, 5},
{B4, 432, 229},
{C5, 225, 0},
{B4, 223, 10},
{A4, 955, 0},
};

void playMidi(int pin, const int notes[][3], size_t len){
for (int i = 0; i < len; i++) {
    tone(pin, pgm_read_word(&notes[i][0]));
    delay(pgm_read_word(&notes[i][1]));
    noTone(pin);
    delay(pgm_read_word(&notes[i][2]));
}
}

void setup() {
    // put your setup code here, to run once:
    Serial.begin(115200);
    Wire.begin();
    Serial.println(F("Serial Communications enabled"));
    if (!DUT_setup()) {
        Serial.println(F("Error in dac or adc initialization"));
    }
    if (!initializeDisplay()){
        Serial.println(F("Error in OLED initialization..."));
        while(1);
    }
    else Serial.println(F("OLED ready."));
    clearScreen();
}

```

```

textAt(0, 20, 3, "Welcome");
disp();
if (!initialise_gyro()) {
    Serial.println(F("Error in gyro initialization..."));
    while(1);
}
buzzer_setup();
smartled_setup();
setAllLeds(0,255, 0);
delay(500);
clearScreen();
disp();
}
int lit_led = 0;
bool test_voltage = false;
bool silentMode = true;

void loop() {
    if (test_voltage) {
        float r = test_device();
        textAt(80,0,2,"RTH");
        textAt(80,24,1,"avg");
        clearSection(80,40,40,20);
        textAt(80,40,1,String(r,3).c_str());
        disp();
        test_voltage = false;
        return;
    }
    movementLoop();
    if (buttonIsPressed(BUTTONA)) { // if button A is pressed
        while (buttonIsPressed(BUTTONA)); // wait for it to not be pressed
        if (buttonIsPressed(BUTTONB)) {
            while(buttonIsPressed(BUTTONB));
            playMidi(BUZZER, midi1, 611);
        }
        test_voltage = true; // schedule the DUT loop to run.
    }
    if (buttonIsPressed(BUTTONB)) {
        while (buttonIsPressed(BUTTONB));
        silentMode = !silentMode;
    }
}

```

```

}

void movementLoop() {
    bool temp = moving;
    PollReturn p = pollGyro();
    moving = p.moving;
    if (temp != moving) {
        clearSection(0,0,64,21);
        if (moving) {
            if (!silentMode) tone(BUZZER, 850);
            text_moving();
            lit_led = 0;
        }
        else text_still();
    }
    if (moving) {
        clearAllLeds();
        setOneLed(lit_led, 255, 0, 255);
        lit_led = (lit_led + 1) % 4;
    }
    else {
        setAllLeds(0,255, 0);
        noTone(BUZZER);
    }
    disp();
    delay(10);
}

```

dut.ino

```

#include <Wire.h>
#include <Adafruit_ADS1X15.h>

#define npts 20 //number of points to measure
#define I_max_A 0.25 // max current to set for
#define I_step_A (float) (I_max_A / npts) //step current change
#define DAC_ADU_per_v (4095.0 / 5.0) //conversion from volts to ADU
#define ADC_V_per_ADU (0.125 * 1e-3) // the voltage of one bit on the gain of 1 scale
#define onTime 1001 // msec

#define RSENSE 10

float V_VRM_loaded = 0.0; // VRM voltage in volts
float V_VRM_thevenin = 0.0; // VRM voltage in volts

```

```

float I_sense_on_total = 0.0; // I_sense
float I_sense_off_total = 0.0;

void dac_write(uint16_t value) {
    Wire.beginTransaction(0x60);
    Wire.write(0x40); // write DAC command byte
    Wire.write(value >> 4); // upper 8 bits
    Wire.write((value & 0xF) << 4); // lower 4 bits
    Wire.endTransmission();
}

bool DUT_setup() {
    ads.setGain(GAIN_ONE); // 1x gain +/- 4.096V 1 bit = 2mV 0.125mV
    if (ads.begin()) Serial.println(F("adc ready.)); // note- you can put the address of
the ADS111 here if needed
    else return false;
    ads.setDataRate(RATE_ADS1115_860SPS); // sets the ADS1115 for higher speed
    Wire.beginTransaction(0x60);
    if (Wire.endTransmission() == 0) Serial.println(F("dac ready.));
    else return false;
    dac_write(0);

    return true;
}

void func_meas_on(float current){
    int write_to_dac = current * RSENSE * DAC_ADU_per_v;
    V_VRM_loaded = 0.0;
    I_sense_on_total = 0.0;

    int counter = 0;
    long stop_time = micros() + (onTime*1000);
    dac_write(write_to_dac);
    while (micros() <= stop_time) {
        V_VRM_loaded = ads.readADC_Differential_0_1() * ADC_V_per_ADU * 3 + V_VRM_loaded;
        I_sense_on_total = ads.readADC_Differential_2_3() * ADC_V_per_ADU / RSENSE +
I_sense_on_total;
        counter++;
    }
    dac_write(0);
    V_VRM_loaded = V_VRM_loaded / counter;
}

```

```

    I_sense_on_total = I_sense_on_total / counter;
}

void func_meas_off(){
    dac_write(0);
    V_VRM_thevenin = 0.0;
    I_sense_off_total = 0.0;
    int counter = 0;
    long stop_time = micros() + onTime*1000*10;
    while (micros() <= stop_time) {
        V_VRM_thevenin = ads.readADC_Differential_0_1() * ADC_V_per_ADU * 3 +
V_VRM_thevenin;
        I_sense_off_total = ads.readADC_Differential_2_3() * ADC_V_per_ADU / RSENSE +
I_sense_off_total;
        counter++;
    }
    V_VRM_thevenin = V_VRM_thevenin / counter;
    I_sense_off_total = I_sense_off_total / counter;
}

float test_device() {
    float Rth_avg = 0.0;
    for (int pt = 1; pt <= npts; pt++) {
        float current = pt * I_step_A;
        dac_write(0);
        func_meas_off();
        func_meas_on(current);
        dac_write(0);
        float Iload = I_sense_on_total - I_sense_off_total;
        float Rth = (V_VRM_thevenin - V_VRM_loaded) / Iload; // calculate thevenin
resistance
        Rth_avg=(Rth > 0) ? Rth_avg + Rth : Rth_avg;
        if (V_VRM_loaded < 0.75 * V_VRM_thevenin) pt = npts;
        Serial.print(pt); Serial.print(F(", "));
        Serial.print(Iload * 1e3, 3); Serial.print(F(", "));
        Serial.print(V_VRM_thevenin, 4); Serial.print(F(", "));
        Serial.print(V_VRM_loaded, 4); Serial.print(F(", "));
        Serial.println(Rth, 4);
    }
    return Rth_avg / 20.0;
}

```

gyro.ino

```
#include "MPU6050.h"

#define GYRO_VERBOSE 1

MPU6050 mpu;

#define ACCEL_THRESHOLD 0.01f // g

// How many consecutive "active" samples must occur before movement is
#define SUSTAIN_SAMPLES 3

// Sample interval
#define SAMPLE_MS 20

// — State —
float baseMagnitude;
int sustainedMovement = 0;
int sustainedStillness = 0;
bool nowMoving = false;
bool movementActive = false;

bool initialise_gyro() {
    Wire.begin();
    mpu.initialize();

    if (!mpu.testConnection()) {
        return false;
    }

    const int CAL_N = 100;
    double sumX = 0, sumY = 0, sumZ = 0;
    // mpu.CalibrateGyro(10);
    for (int i = 0; i < CAL_N; i++) {
        int16_t ax, ay, az, gx, gy, gz;
        mpu.getMotion6(&ax, &ay, &az, &gx, &gy, &gz);
        // Convert raw to g (AFS_SEL default = 0 -> +/-2 g -> 16384 LSB/g)
        sumX += ax / 32768.0f;
        sumY += ay / 32768.0f;
        sumZ += az / 32768.0f;
        delay(SAMPLE_MS);
    }
}
```

```

}

float baseAx = sumX / CAL_N;
float baseAy = sumY / CAL_N;
float baseAz = sumZ / CAL_N;

baseMagnitude = sqrt(baseAx*baseAx + baseAy*baseAy + baseAz*baseAz); // get the
baseline magnitude of the device while standing still
Serial.print(F("Baseline Magnitude=")); Serial.println(baseMagnitude, 3);
Serial.println(F("Gyro ready.));
return true;
}

PollReturn pollGyro() {
    int16_t ax, ay, az, gx, gy, gz;
    mpu.getMotion6(&ax, &ay, &az, &gx, &gy, &gz);

    // Convert raw to g
    float accelX = ax / 32768.0f;
    float accelY = ay / 32768.0f;
    float accelZ = az / 32768.0f;

    float magnitude = sqrt(accelX*accelX + accelY*accelY + accelZ*accelZ); // get current
total acceleration on the gyroscope (should theoretically be around 1g when still)
    // Serial.print(F("magnitude=")); Serial.println(magnitude, 3);

    // make sure that the change is sustained - don't care about small moments of
stillness or movement... ex. if the board for some reason is at the perfect baseline g
sometime while someone is moving it in the air, we want it to still think it is moving

    if (magnitude - baseMagnitude > ACCEL_THRESHOLD || baseMagnitude - magnitude >
ACCEL_THRESHOLD) sustainedMovement++;
    else sustainedMovement = 0;
    if (magnitude - baseMagnitude < ACCEL_THRESHOLD && baseMagnitude - magnitude <
ACCEL_THRESHOLD) sustainedStillness++;
    else sustainedStillness = 0;

    if (sustainedMovement >= SUSTAIN_SAMPLES) nowMoving = true;
    else if (sustainedStillness >= SUSTAIN_SAMPLES) nowMoving = false;

```

```
// — Edge detection: report transitions only —————
if (nowMoving && !movementActive) {
    movementActive = true;
} else if (!nowMoving && movementActive) {
    movementActive = false;
}
PollReturn ret = {movementActive, magnitude};
return ret;
}
```

oled.ino

```
#include <Wire.h>
#include <Adafruit_GFX.h>
#include <Adafruit_SSD1306.h>

#define SCREEN_WIDTH 128
#define SCREEN_HEIGHT 64
#define OLED_ADDRESS 0x3C

Adafruit_SSD1306 display(SCREEN_WIDTH, SCREEN_HEIGHT);

bool initializeDisplay() {
    return display.begin(SSD1306_SWITCHCAPVCC, OLED_ADDRESS);
}

bool testinit() {
    Serial.println(F("Starting OLED init..."));
    // Check the OLED is still responding
    Wire.beginTransmission(0x3C);
    uint8_t error = Wire.endTransmission();
    Serial.print(F("OLED ACK before begin(): "));
    Serial.println(error); // 0 = device responded, anything else = problem
    bool result = display.begin(SSD1306_EXTERNALVCC, OLED_ADDRESS);
    Serial.print(F("display.begin() returned: "));
    Serial.println(result);
    // Check again after
    Wire.beginTransmission(0x3C);
    error = Wire.endTransmission();
    Serial.print(F("OLED ACK after begin(): "));
    Serial.println(error);
    return result;
}
```

```

void clearScreen() {
    display.clearDisplay();
    display.setTextColor(SSD1306_WHITE);
    display.setCursor(0, 0);
}

void textAt(int x, int y, uint8_t size, const char *text) {
    display.setTextSize(size);
    display.setCursor(x, y);
    display.print(text);
}

void disp() {
    display.display();
}

void clearSection(int x, int y, uint16_t width, uint16_t height) {
    for (int i = 0; i < height; i++) {
        display.drawFastHLine(x, y+i, width, SSD1306_BLACK);
    }
}

void text_moving(){
    textAt(0,0,2,"Move");
}

void text_still(){
    textAt(0,0,2,"Still");
}

```

io.ino

```

#include <Adafruit_NeoPixel.h>

#define SMART_LED 9
#define BUZZER 6

Adafruit_NeoPixel strip(4, SMART_LED, NEO_GRB + NEO_KHZ800);

void smartled_setup(){
    strip.begin();
    strip.show();
    strip.setBrightness(20);
}

```

```

}

void buzzer_setup() {
    pinMode(BUZZER, OUTPUT);

    // Example: short beep at 1kHz
    tone(BUZZER, 1000);
    delay(100);
    noTone(BUZZER);
    delay(50);
    tone(BUZZER, 1000);    // A4
    delay(100);
    noTone(BUZZER);
}

void setAllLeds(uint8_t r, uint8_t g, uint8_t b) {
    for (int i = 0; i < 4; i++) {
        strip.setPixelColor(i, strip.Color(r, g, b));
    }
    strip.show();
}

void setOneLed(uint8_t index, uint8_t r, uint8_t g, uint8_t b) {
    if (index >= 4) return;
    strip.setPixelColor(index, strip.Color(r, g, b));
    strip.show();
}

void clearAllLeds() {
    strip.clear();
    strip.show();
}

void button_setup() {
    pinMode(BUTTONA, INPUT);
    pinMode(BUTTONB, INPUT);
}

bool buttonIsPressed(int button) {
    if (!digitalRead(button)) return true;
    return false;
}

```