Board 4 Report:
A 4-layer Instrument Droid Board

Andrew Thibeault
ECEN 5730
CU Boulder

2023-05-07

Plan of Record

This board is designed to be a 4-layer instrument droid board, with a microcontroller and data acquisition system. The microcontroller will be bootloaded as an Arduino Uno, and header pins in standard Arduino Uno configuration will be made available to interface with the microcontroller pins outside of its instrument droid capabilities. The instrument droid is designed to measure Thevenin output resistance of any supply as the current load of the supply changes, with the data available for acquisition via usb connection to a computer running code with an Arduino API.

The main focuses of this board are to practice using a 4-layer design in order to optimize performance by minimizing cross-unders that create gaps in the ground plane, as well as demonstrating the instrument droid which will reveal the limitations of power sources inherent in their Thevenin resistance. The Arduino connectivity designed in Board 3 will be reused, with the instrument droid added onto this.

The layer stack-up will be designed as Signal | Ground Plane | Ground Plane | Signal. This stack up allows connection of return currents between planes with a simple through-hole via. A through-hole via will be included next to every signal via that passes from the top layer to the bottom layer. This stack-up also allows signal traces to change planes to another signal layer to avoid cross-unders that create breaks in a return plane.

The general instrument droid design follows:

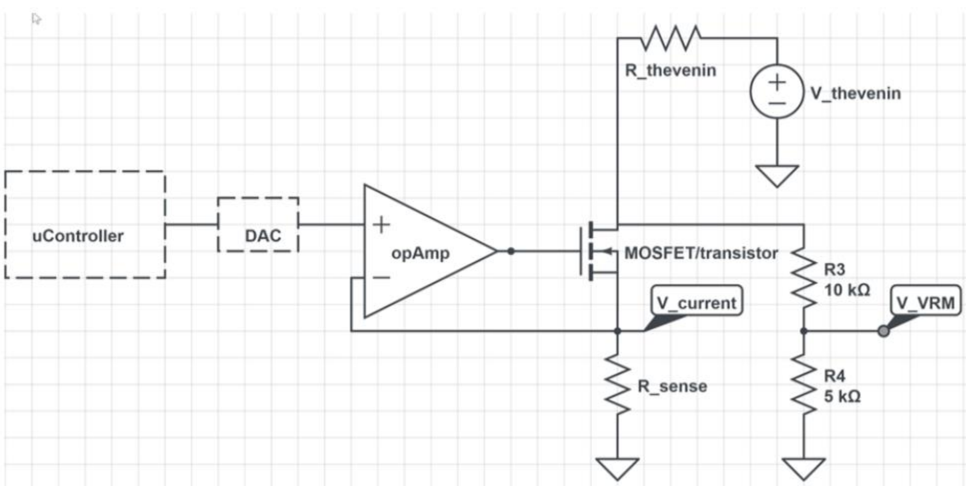

Figure 1: Thevenin Resistance

The input from the voltage source to the board will be by USB mini, power jack, or screw terminal, and switches will be included to only allow one of these sources to be connected at a time. The input voltage will be up to 12 V. The Thevenin voltage is measured at the V_VRM testpoint. A voltage divider is employed in order to meet the input voltage specs of an ADS1115, which will be used to measure the differential voltage with respect to local ground at V_current and V_VRM. In order to measure Thevenin resistance, the microcontroller first instructs the

DAC to output a voltage pulse. This causes the opAmp to output a voltage that turns on the MOSFET just enough to obtain the current through R_sense that produces the same voltage at the inverting opAmp input as output by the DAC. Different voltage pulses (different load currents) will be tested to determine the Thevenin resistance of the source at different loads. A switch will be included to choose a 1 ohm or 10 ohm resistance for R_sense.

There are several design constraints that will be implemented in the code used to drive the DAC. The load current (DAC voltage) will be pulsed at 10% duty cycle in order to avoid overheating any parts. The current will ramp in 20 steps from 0 A to 300 mA. The current will stop if the loaded voltage on the VRM drops below 75% of the unloaded voltage.

10k ohm pullup resistors will be included on the scl and sda lines. The Atmega 328 already has internal pullup resistors on these lines, but these external resistors are included for risk mitigation. The lower external pullup resistance will ensure that the time constant of the rising edge associated with pulling the line high (writing a 1-bit) is short enough for the devices on the I2C bus to communicate properly at their specified data rate.

Finally, RGB smart LEDs and a simple buzzer will be included on the board to indicate when current is flowing from the VRM under test. These will be hardwired to digital pins on the microcontroller, and controlled using the instrument droid code.

The final schematics and board layout are shown on the following pages.

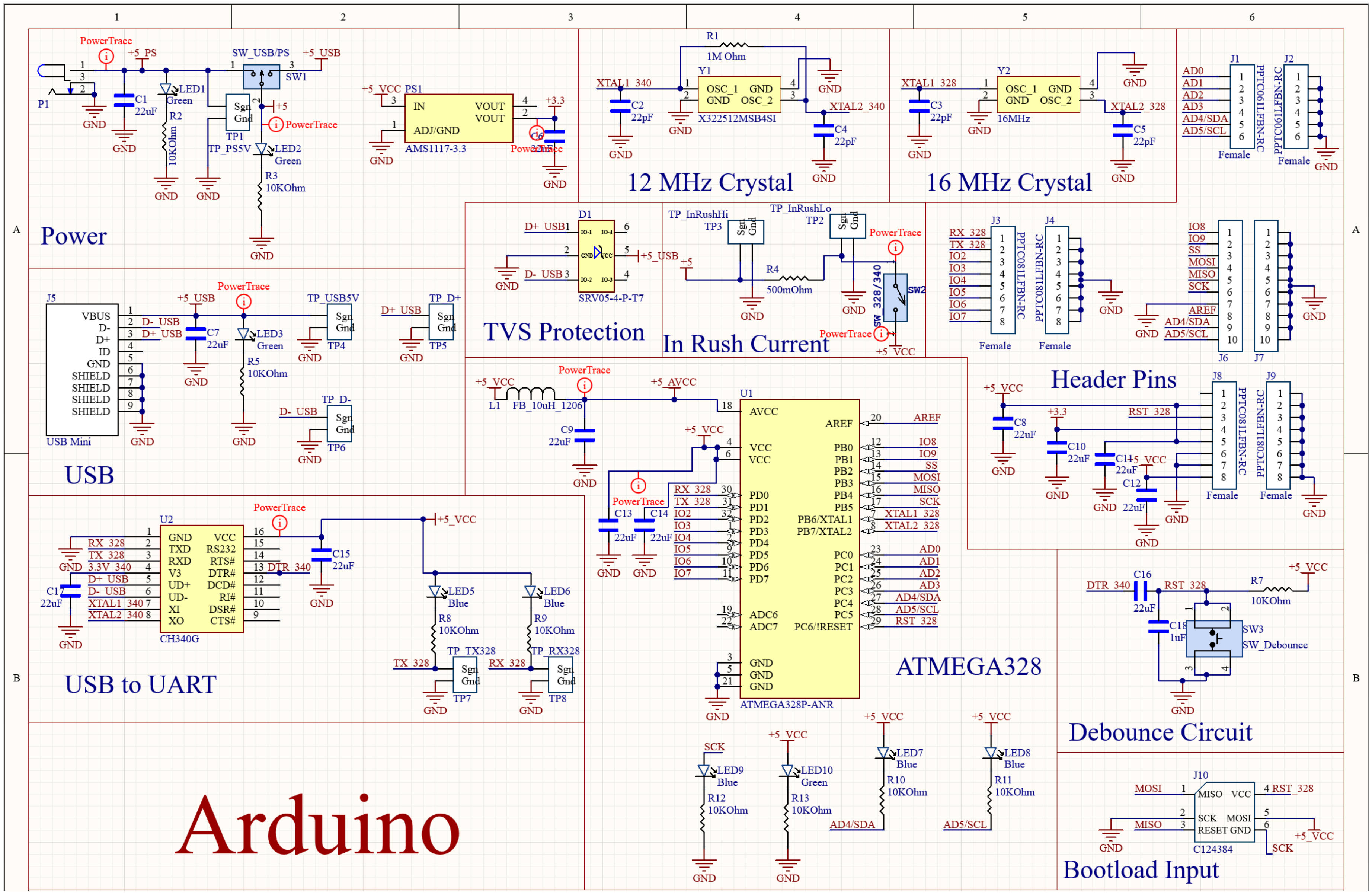

Figure 2: Arduino Schematic

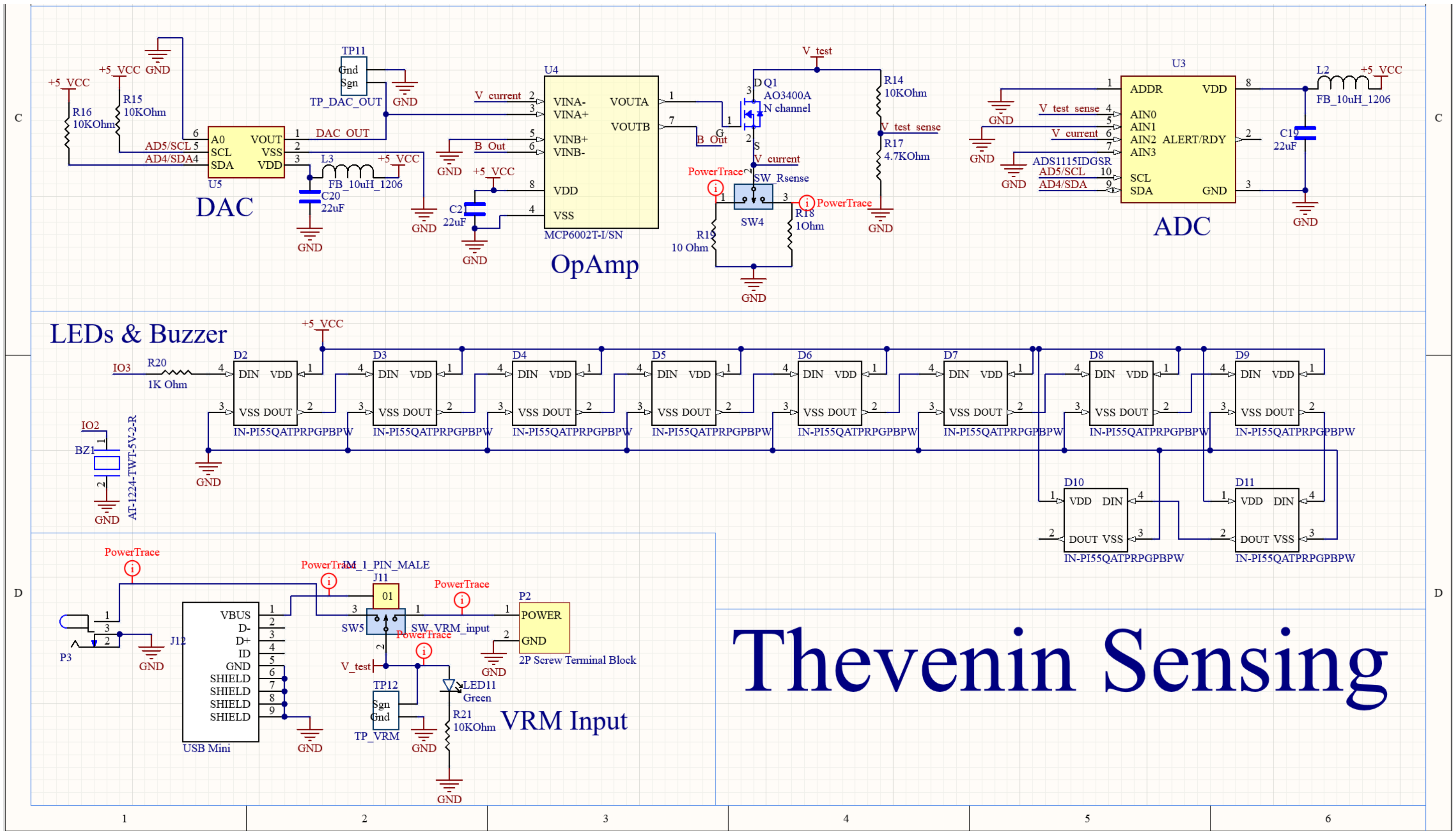

Figure 3: Instrument Droid Schematic

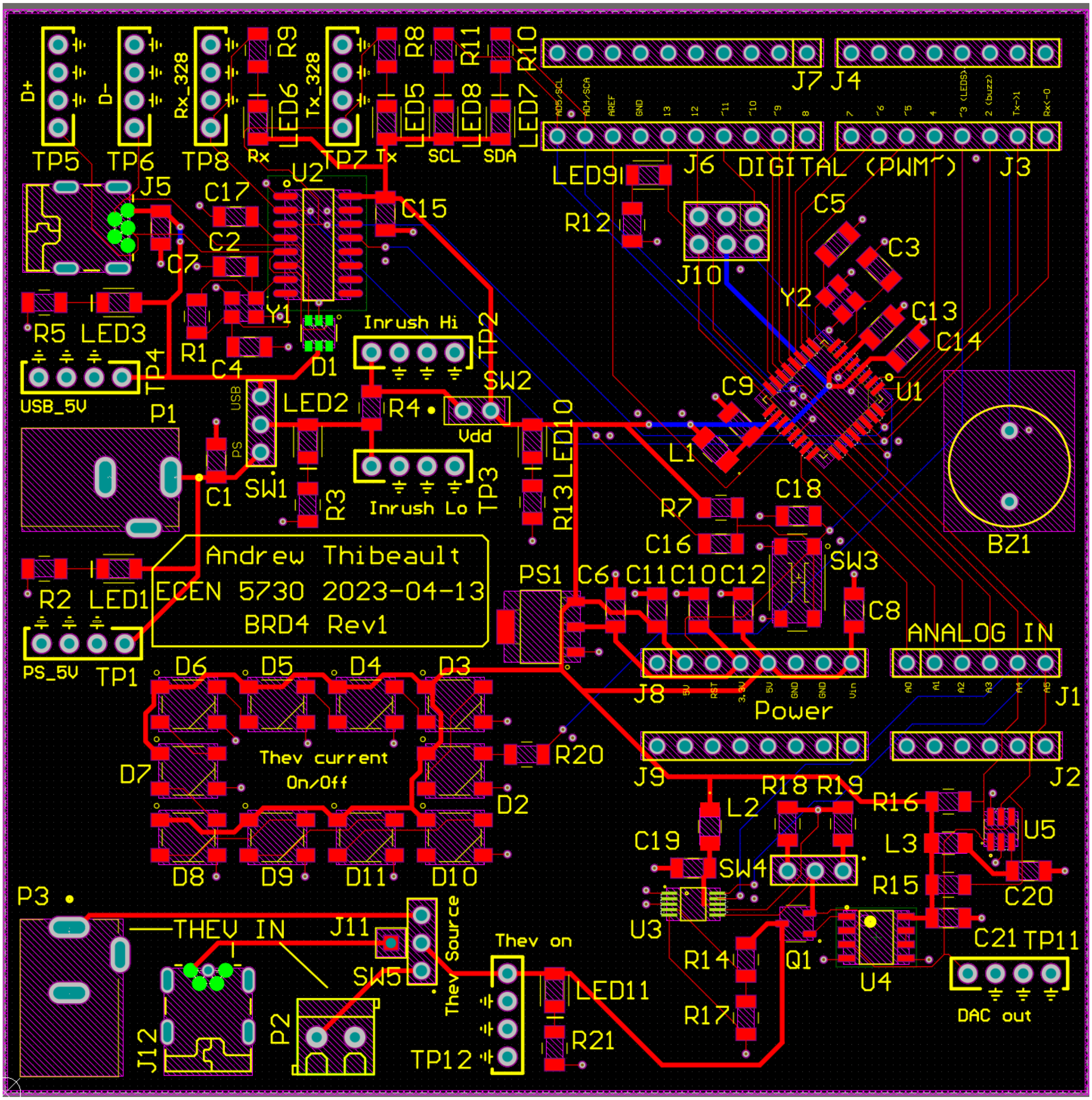

Figure 4: Board 4 Layout

(Signal layers in red and blue, ground plane layers not shown for ease of viewing traces)

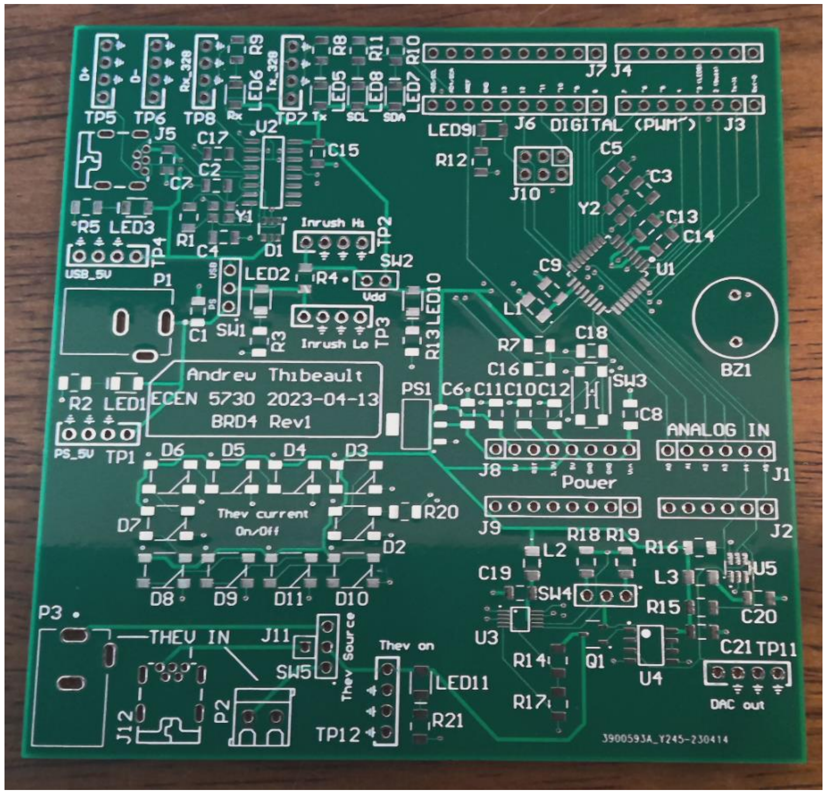

Figure 5: Front of bare board

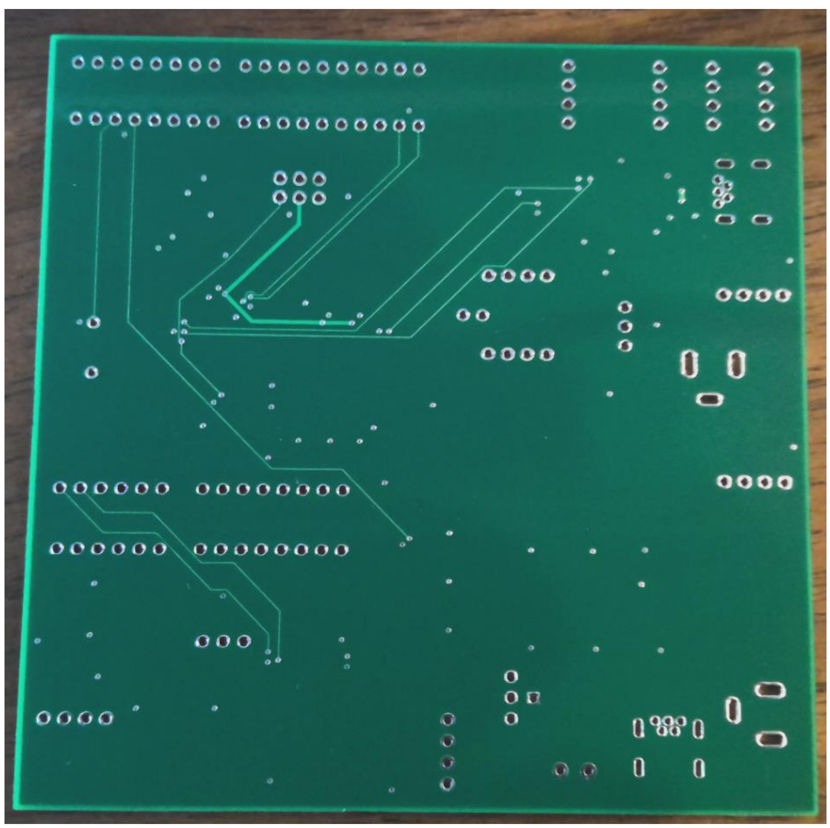
Figure 6: Back of bare board

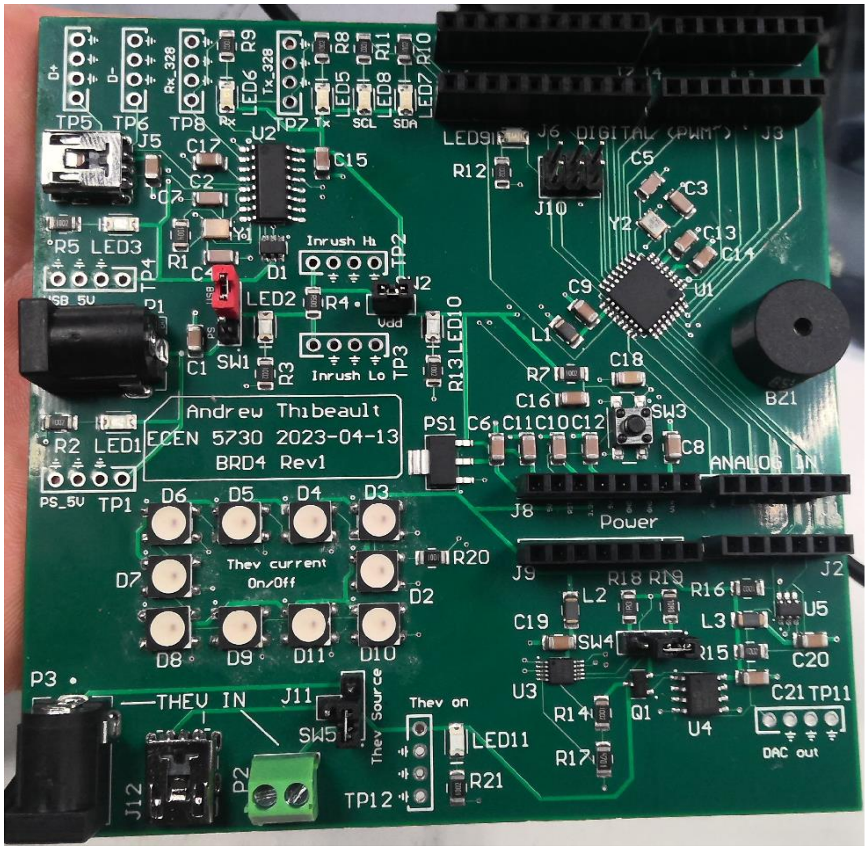

Figure 7: Fully Assembled Board 4

Board Outcomes

Assembly of this board took a significant amount of time due to the number of components, and I did not fully complete self assembly due to receiving the fully assembled board from the fab shop. In the future, for a more streamlined instrument droid, I would not include irrelevant Board 3 components in order to make assembly easier, such as the Arduino header pins, the 3.3 V VRM, inrush current circuitry, and the decoupling capacitors for Arduino power pins. I also used a larger board for Board 4, which allowed more breathing room for components and routing that was previously more congested. This allowed for easier assembly of some components that were previously spaced very closely.

The usage of 4 layers on this board allowed for all signal routing to occur only on the signal layers – no cross-unders on either of the ground planes were used. This is in comparison to Board 3, in which more than 10 cross-unders were utilized, with some being relatively long, longer than 200 mil.

There were several flaws with this board. I did not include enough test points around the new instrument droid circuit, and frequently had to probe directly on pads to get the measurements I needed. I also neglected to include labeling on the bootload pins despite changing them from the standard configuration in order to improve routing. This was an issue when bootloading, as I had to refer back to the Altium layout to wire the pins correctly.

The Atmega 328 successfully bootloaded and ran the simple blink sketch. I included an indicator LED on pin 13 in this board to comply with the LED_BUILTIN LED pin for the Arduino Uno. I did not include this LED in Board 3, which made testing the Blink sketch slightly confusing as I expected to see an LED lighting.

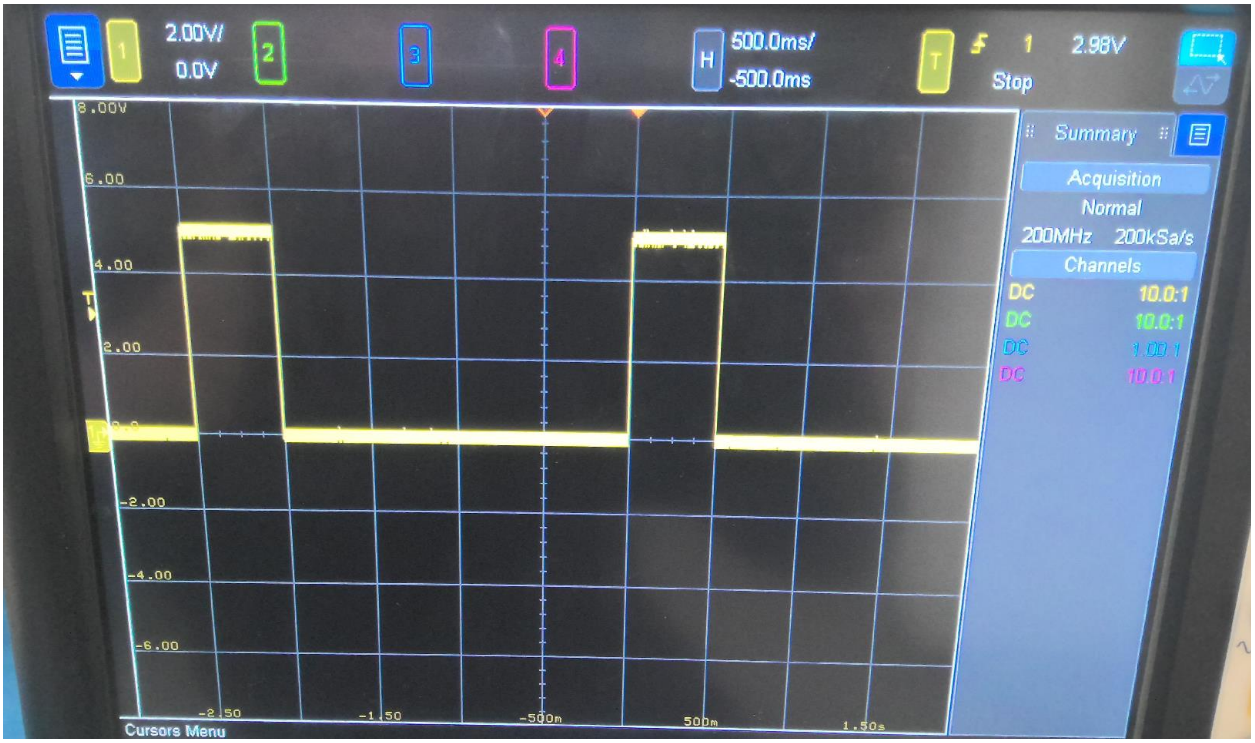

Figure 8: Scope trace of pin 13 (LED_BUILTIN pin) with modified Blink code running 20% duty cycle, 2.5 s period

Once the Atmega 328 was successfully bootloaded as an Arduino Uno and demonstrated that it could run Arduino code, I performed the comparative noise analysis that was used for Board 3. The results of the noise analysis are below:

Board	V_on 63 ohm resistor (TP3)	Estimated Total Aggressor Current	Rise Time	Fall Time	Q_LO Noise, Rising Edge	Q_LO Noise, Falling Edge	Q_HI Noise, Rising Edge	Q_HI Noise, Falling Edge	Power Rail Noise, Rising Edge	Power Rail Noise, Rising Edge
Keyestudio Arduino Uno	1.59 V	77 mA	7.0 ns	6.7 ns	277 mV	683 mV	387 mV	393 mV	32 mV	53 mV
Board 3	1.92 V	93 mA	5.5 ns	6.1 ns	249 mV	572 mV	333 mV	316 mV	30 mV	45 mV
Board 4	1.70 V	81 mA	6.0 ns	6.7 ns	152 mV	501 mV	223 mV	235 mV	21.7 mV	40.2 mV

Table 1: Noise Measurements with Atmega 328 as Aggressor Signal

Scenario	Noise as Percent of Commercial Board Noise					
Scenario	Q_LO Noise, Rising Edge	Q_LO Noise, Falling Edge	Q_HI Noise, Rising Edge	Q_HI Noise, Falling Edge	Power Rail Noise, Rising Edge	Power Rail Noise, Rising Edge
Board 3 Measured	89.9 %	83.7 %	86.0 %	80.4 %	93.8 %	84.9 %
Board 4	54.9 %	73.3 %	57.6 %	59.8 %	67.8 %	75.8 %

Table 2: Board 3 Noise as Percent of Commercial Board Noise with Atmega 328 as Aggressor Signal

Board	V_on 10 ohm resistor	Estimated Total Aggressor Current	Rise Time	Fall Time	Q_HI Noise, Rising Edge	Q_HI Noise, Falling Edge	Power Rail Noise, Rising Edge	Power Rail Noise, Rising Edge
Keyestudio Arduino Uno	3.5 V	350 mA	20.0 ns	8.0 ns	443 mV	253 mV	480 mV	720 mV
Board 3	3.65 V	365 mA	19.1 ns	7.6 ns	384 mV	258 mV	531 mV	730 mV
Board 4	3.55 V	355 mA	21.0 ns	8.0 ns	102 mV	76.9 mV	366 mV	594 mV

Table 3: Noise Measurements with board as Aggressor Signal

Scenario	Noise as Percent of Commercial Board Noise			
Scenario	Q_HI Noise, Rising Edge	Q_HI Noise, Falling Edge	Power Rail Noise, Rising Edge	Power Rail Noise, Rising Edge
Board 3 Measured	86.7 %	102.0 %	110.6%	101.4%
Board 4	23.0 %	30.4 %	76.3 %	82.5 %

Table 4: Board 3 Noise as Percent of Commercial Board Noise with Atmega 328 as Aggressor Signal

As can be seen from Table 2 and Table 4, the noise improvement from the commercial Arduino is dramatic in every scenario, and noise is greatly improved from Board 3 as well. Noise is reduced on Board 4 by the use of decoupling capacitors and continuous ground planes. Improvements from Board 3 are likely due to eliminating signal cross-unders on the ground plane, as well as shorter signal traces from improved board organization and the use of two signal layers for routing. In several cases on Board 3, signal traces were long to avoid the use of cross-unders. On Board 4, this was avoided by the use of the second signal layer.

The DAC was successfully communicated with, and a simple sketch was uploaded to output a pulsating voltage from the DAC.

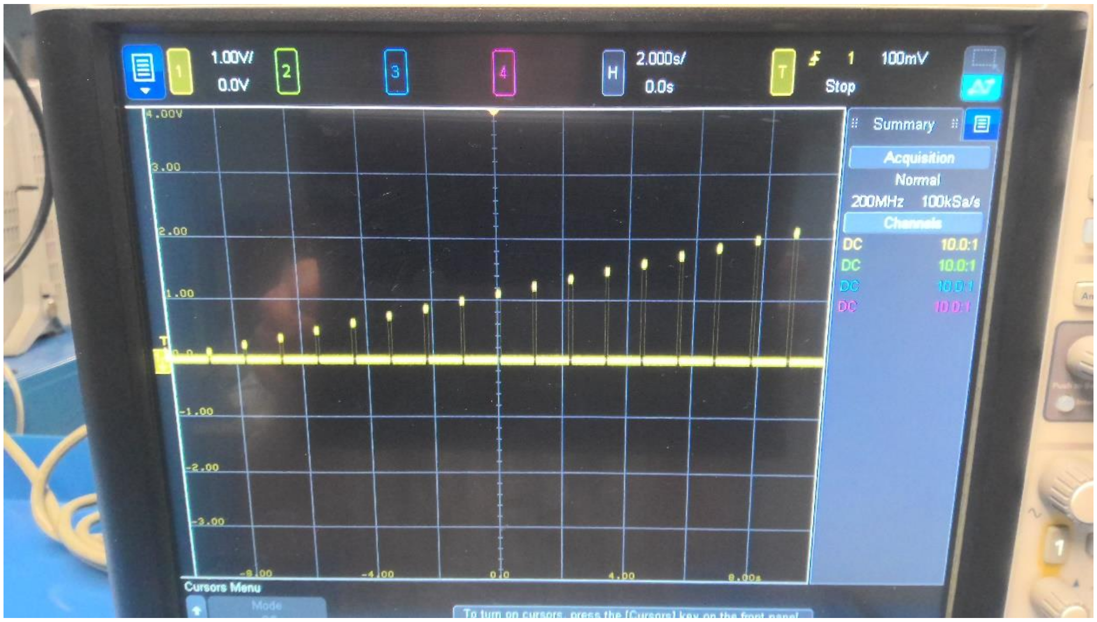

Figure 9: DAC output while running instrument droid code. The pulsed voltage gradually increases, with 10% duty cycle.

The sense resistor successfully exhibits pulsed constant current when the DAC outputs a pulsed voltage. This also indicates that the MOSFET successfully turns on. In order to verify the correct current is passing through the sense resistor, a 9V VRM was connected to the board, and it was attempted to drive 20 mA through the 10 ohm sense resistor. This requires a 200 mV pulse from the DAC, and 200 mV should be seen across the sense resistor. A test point was not included across the sense resistor, so the pads on either end of the resistor were probed instead. A constant 20 mA current was verified.

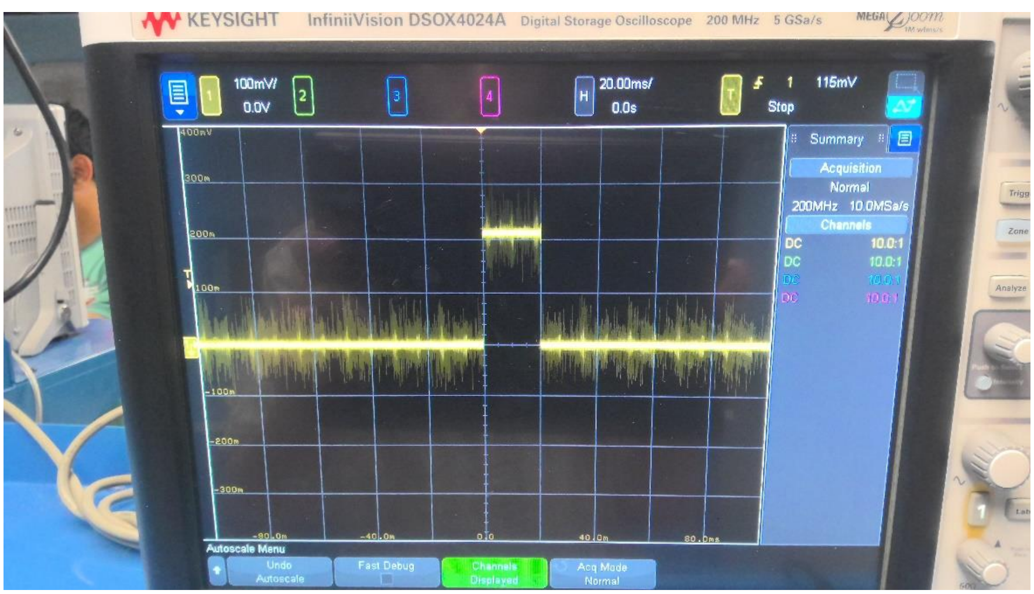

Figure 11: 200 mV pulse through 10 ohm sense resistor (20 mA)

The ADC successfully measures voltage, as can be seen in the serial monitor output when running the instrument droid code. Interestingly, the measured voltage is slightly lower in each case than the input voltage as measured by a probe, even when measuring the voltage directly at the ADC input. This may be due to noise on the board, however a ferrite bead was included at the input to the ADC to filter power rail noise.

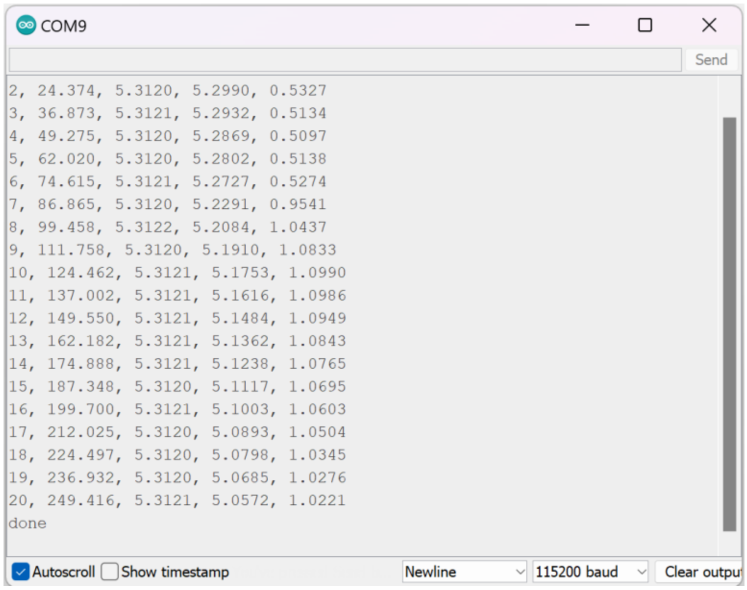

Figure 10: Serial monitor output demonstrating that the ADC successfully measures voltages while running the instrument droid code with a 5 V wall wart input

Code was included to show an indication on the smart LEDs and buzzer of when the code was running and current was being drawn from the VRM. 10 smart LEDs were included on the board. I forgot to include decoupling capacitors next to the LEDs, but they functioned as expected regardless. The LEDs were coded to light up one after another, the first at the start of the loop of voltage pulses and each subsequent LED after every two subsequent pulses (to reach the 10th LED when the 19th and 20th pulses were output from the DAC). The buzzer was configured to sound a tone for the duration of the code sending the voltage pulse from the DAC, or equivalently, for the duration of the current through the sense resistor.

```
void loop() {
  uint16_t c = strip.Color(0, 0, 127);
  for (i = 1; i <= npts; i++) {
    strip.setPixelColor((i-1)/2,c); // Progressively turn on one LED
                                    // For every two voltage pulses

    I_A = i * I_step_A;
    dac.setVoltage(0, false); //sets the output current
    func_meas_off();

    strip.show();
    tone(2,100); // activate buzzer tone
    func_meas_on();
    noTone(2); // turn off buzzer
    dac.setVoltage(0, false); //sets the output current
```
Figure 12: LED and buzzer code included in the instrument droid code

Both of these functionalities worked successfully.

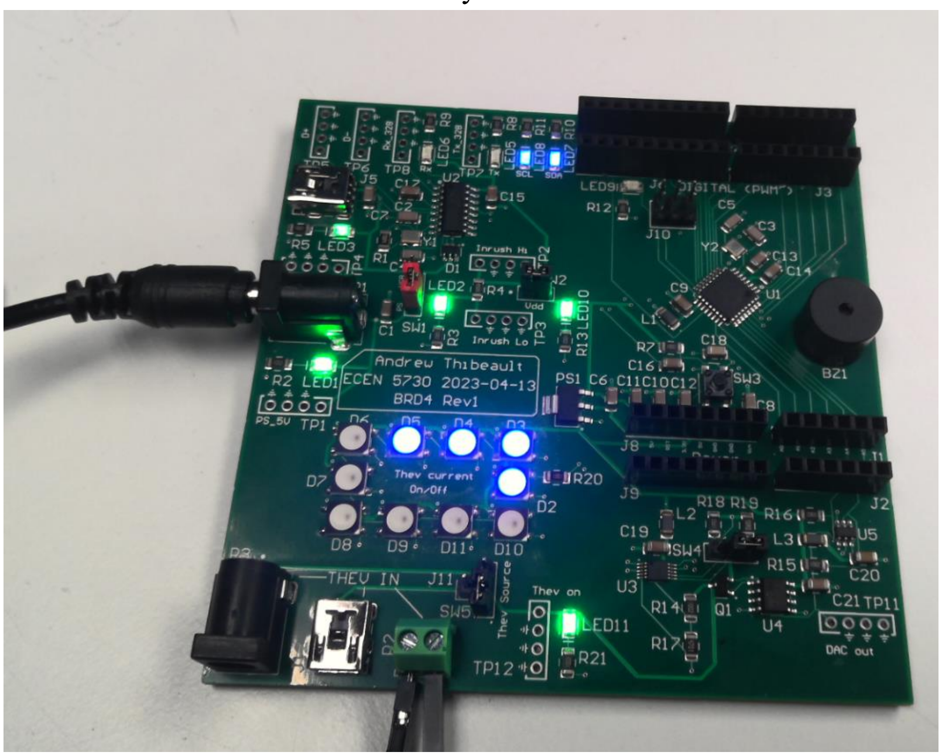

Figure 13: Smart LEDs lighting while running the instrument droid code.
4 lit LEDs indicates the 7th or 8th voltage pulse is being sent from the DAC.

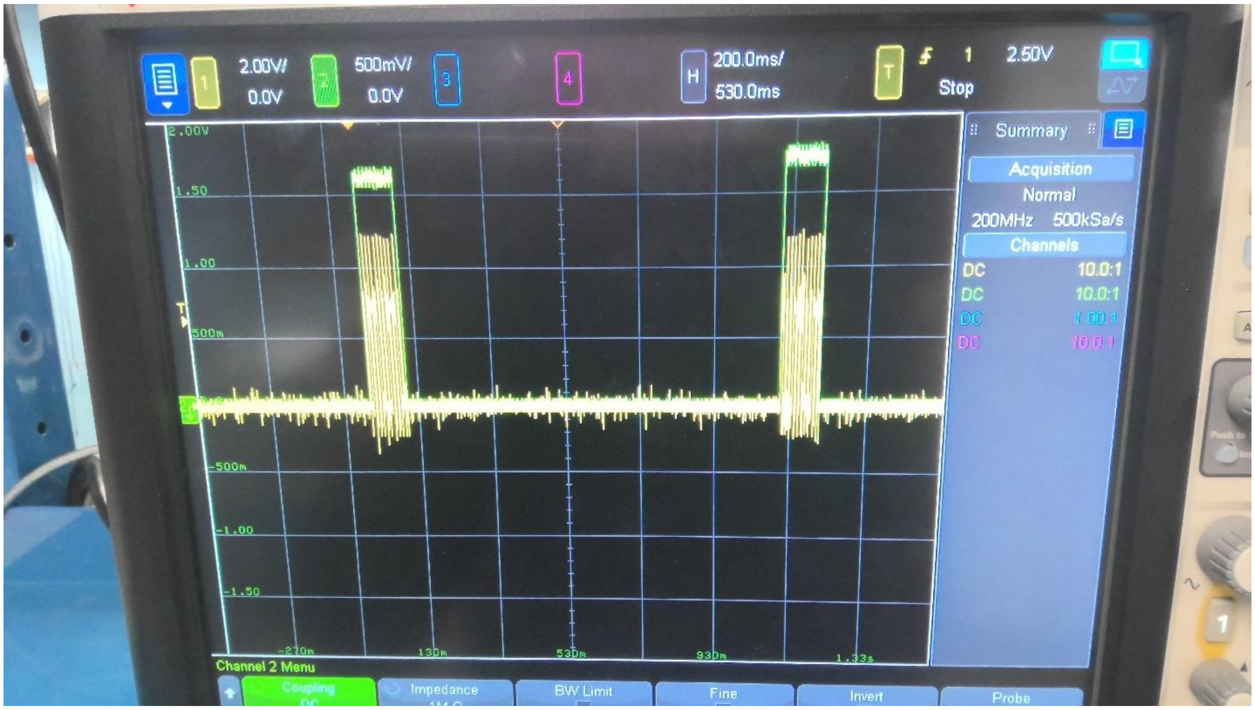

Figure 14: Buzzer pin (Digital pin 2) showing a signal (yellow) being sent to the buzzer whenever the DAC output voltage (green) pulses

Finally, the full instrument droid code was run with various voltage sources connected. Some voltage sources exhibited a very steady R_Thevenin with varying load current, while others exhibited variation. Some examples are shown below.

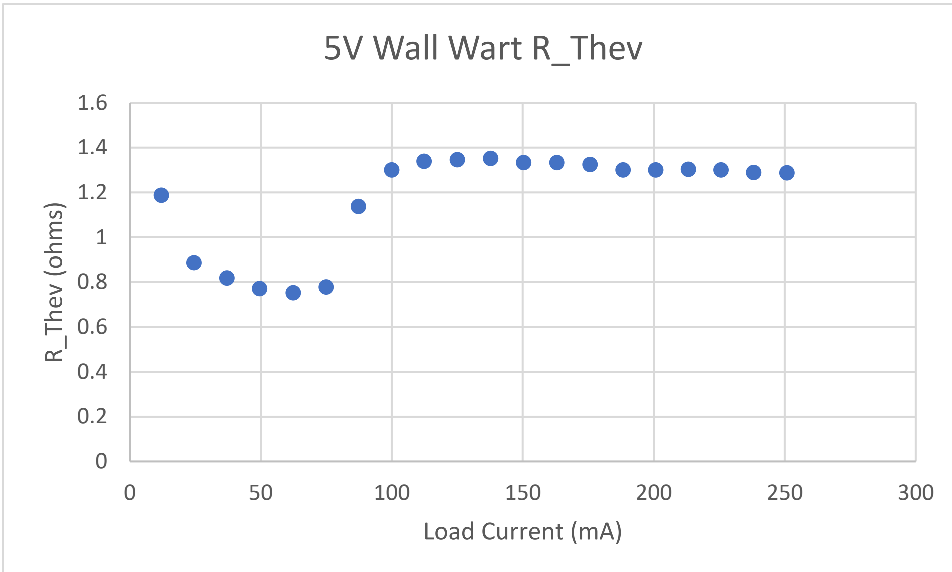

Figure 15: 5 V wall wart R_Thev, displaying some variation with varying load current

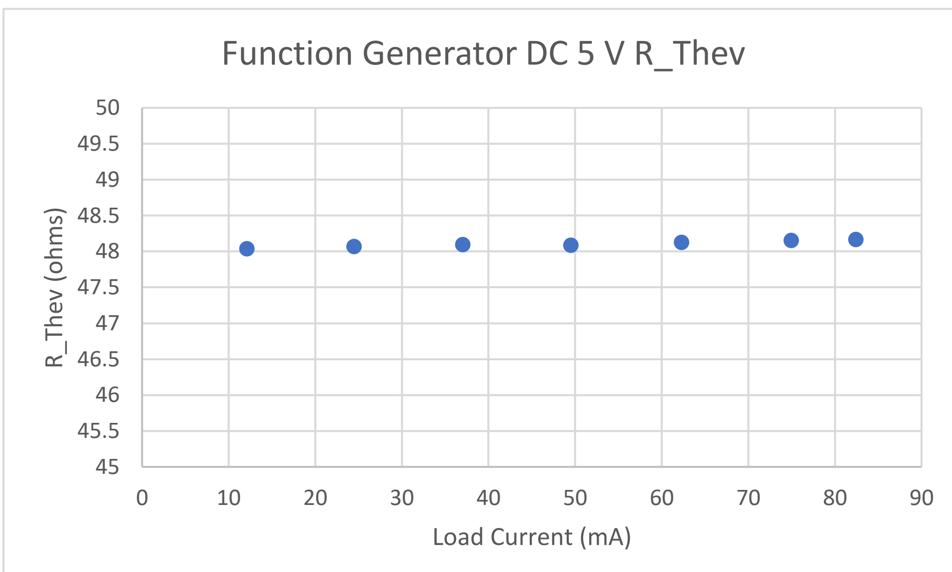

Figure 16: Function generator R_Thev, showing consistent measurements of 48.1 ohms

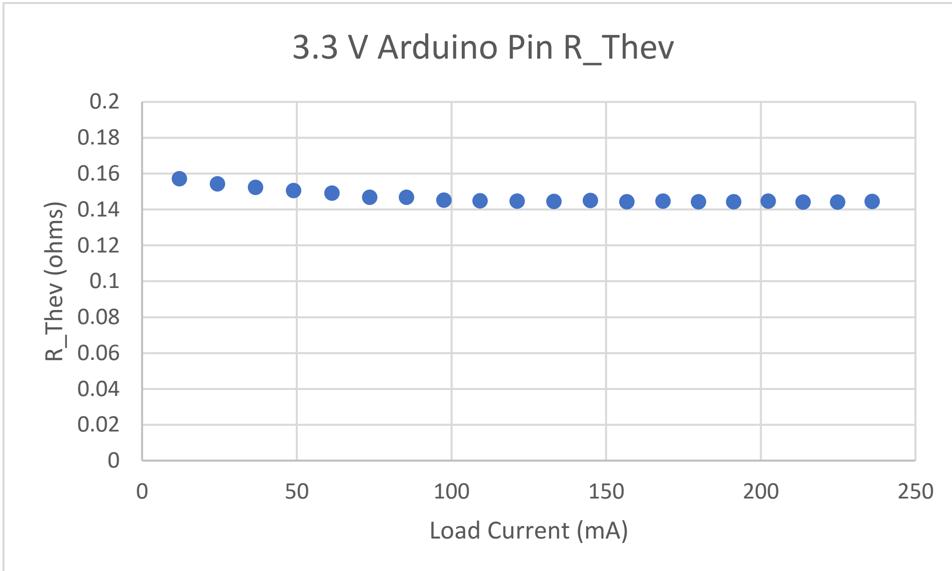

Figure 17: 3.3 V pin on Arduino R_Thev showing relatively consistent measurements. This is likely due to the fact that a stable VRM close to this pin on the board supplies the voltage at this pin.

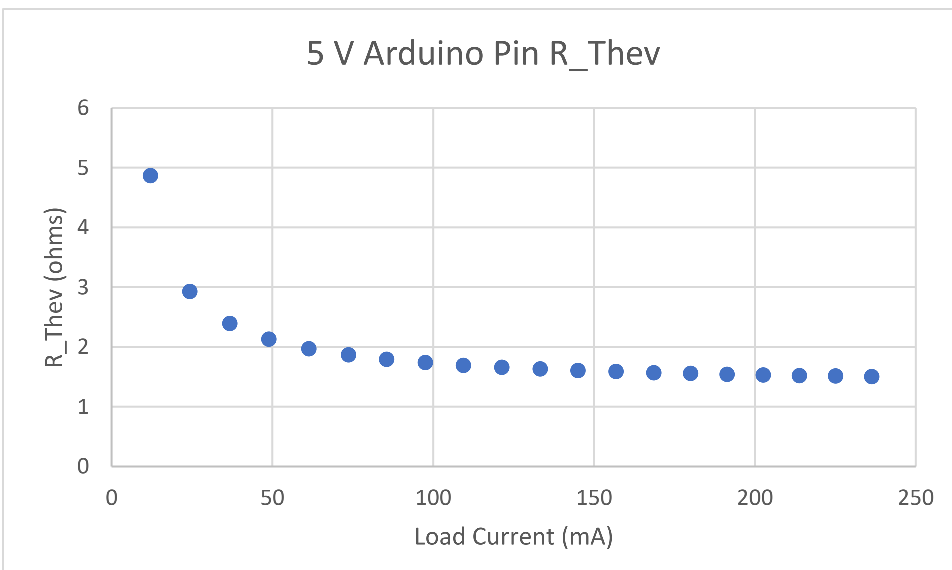

Figure 18: 5 V pin on Arduino R_Thev showing varying R_thev at low load currents. There is no VRM between the USB 5 V pin and this pin, so in reality this is a measurement of the Thevenin resistance of the 5 V pin on the USB, including any impedance on the board between the Arduino pin and the USB pin.

The value of the sense resistance was changed to see the effect on the R_Thevenin measurements. Both values resulted in similar readings, though the 1 ohm R_sense exhibited just slightly more variation in measured values. The 5V power supply exhibited strange variation in both cases at 100 mA of load current, but the cause of this is unknown.

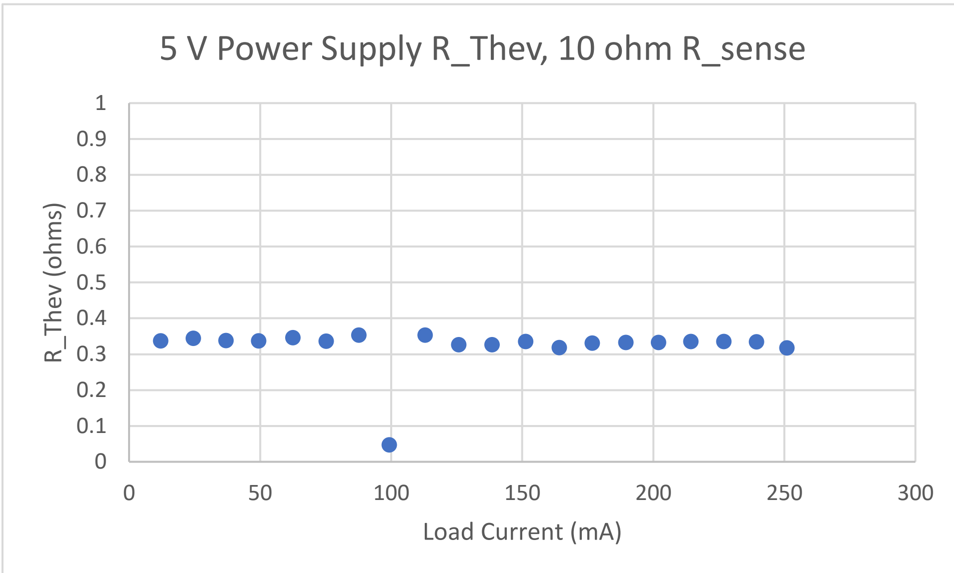

Figure 19: 5 V Power Supply R_Thev using 10 ohm sense resistor. 0.335 ohm average, excluding anomaly at 100 mA

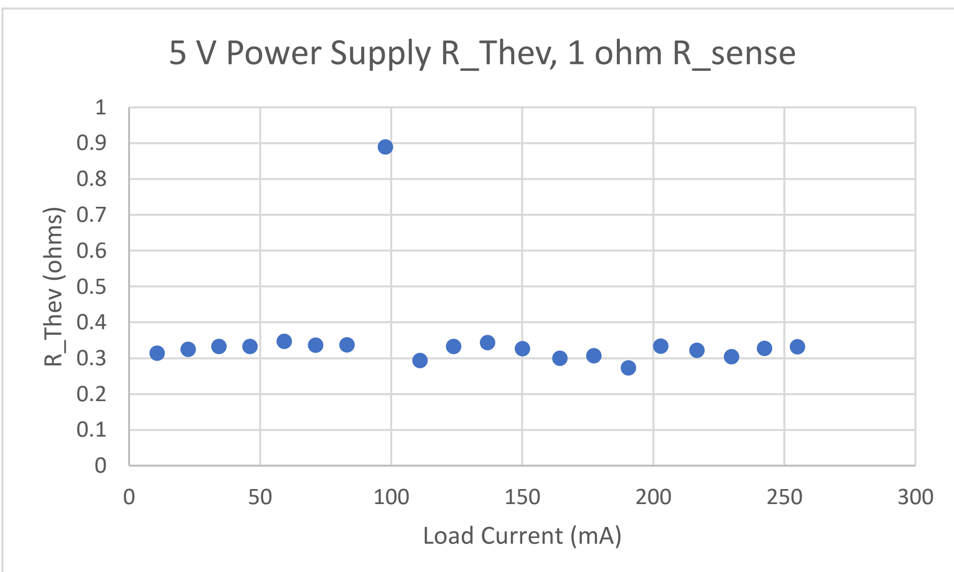

Figure 20: 5 V Power Supply R_Thev using 1 ohm sense resistor. 0.322 ohm average, excluding anomaly at 100 mA

Overall, this board performed its desired function and demonstrated significantly reduced noise characteristics than the previously designed 2-layer Golden Arduino and particularly from the commercial Arduino, with noise reduced mainly to 20-70% of that on the commercial board. The usage of 4 layers allowed for shorter, more spread out traces with no cross-unders into the ground planes. This board also demonstrated the value of prototyping a board and identifying errors. Several aspects were improved from Board 3, such as better decoupling capacitor placement and better orientation of the Atmega 328 to minimize signal trace length. To improve on this board, I would ensure to include decoupling capacitors next to the smart LEDs to ensure their proper function, more test points around the new circuit, and more attention to detail on labeling important pins such as the bootload pins. The instrument droid performed as designed, with the MOSFET turning on and the expected current being seen in the sense resistor given a command to the DAC, as well as the ADS1115 successfully reading the differential voltages needed. With the circuit performing as designed, I was able to successfully characterize the Thevenin resistance of a variety of voltage sources with varying load currents, allowing me to better understand the nature of these voltage sources when using them in the future.