JITHENDRA HS
5/5/24

PCB REPORT – BOARD 4

Instrument Droid (Voltage source Characterizer)

Objective:

Using an Atmega328P microcontroller designed a 4-layer board to characterize the voltage sources by determining the Thevenin Voltage and Thevenin resistance. The best practice like Via's is added near to the trace which is transition from top layer of the board to bottom layer of the board to reduce the noise. Also incorporated OLED display, smart LED's and Buzzer for better user experience.

Working Principle:

- Code reads the Thevenin voltage across the Voltage divider resistor V_{VRM} when there is no gate voltage applied across the MOSFET.
- Code reads the Load Current I_L when pulse voltage applied at MOSFET gate with the help of DAC and opAmp circuit for different current drawn capacity.
- Calculate the Thevenin resistance using formula: $R_{Th} = (V_{Th} – V_L)/I_L$.

Plan of record:

- 5V power input rating from either power jack or USB-mini.
- Sense resistor to measure inrush current.
- 12MHz oscillator for USB data transmission.
- 16MHz oscillator for system clock.
- USB to UART communications to upload sketches.
- Circuit for microcontroller reset.
- Ferrite beads to reduce noise on AVCC.
- Bypass/decoupling capacitors to compensate for the current surge in power rail.
- Low pass filters.
- LED indicators
- Circuit to avoid switch debouncing.
- Jump start circuit for oscillators.
- Display to enhance user experience.

Risk reduction:

- Added LEDs to indicate power supply is working fine.
- ESD protection to USB mini port.
- Proper labelling for each input and outputs.
- Added test points to each module to read measurements and verify the functionality of module.

What does it mean to work:

- Obtaining a 5V power supply from the power jack and USB mini.
- Utilizing a 12MHz frequency oscillator for the USB data transmission.
- Utilizing a 16MHz oscillator frequency for the system clock.
- Data transmission via D+ and D- pins.
- Data transmission via Tx and Rx.
- Implementing proper microcontroller reset without debouncing.
- Ensuring that LED indicators are working fine.

PCB REPORT – BOARD 4

- No signal integrity problem wrt to Via's.
- Able to connect the voltage source through power jack and Screw terminal.
- Able to communicate and modify DAC output.
- Able to communicate and read ADC output.
- Able to measure the Thevenin Voltage and Thevenin resistance.
- Able to display the reading in real time.

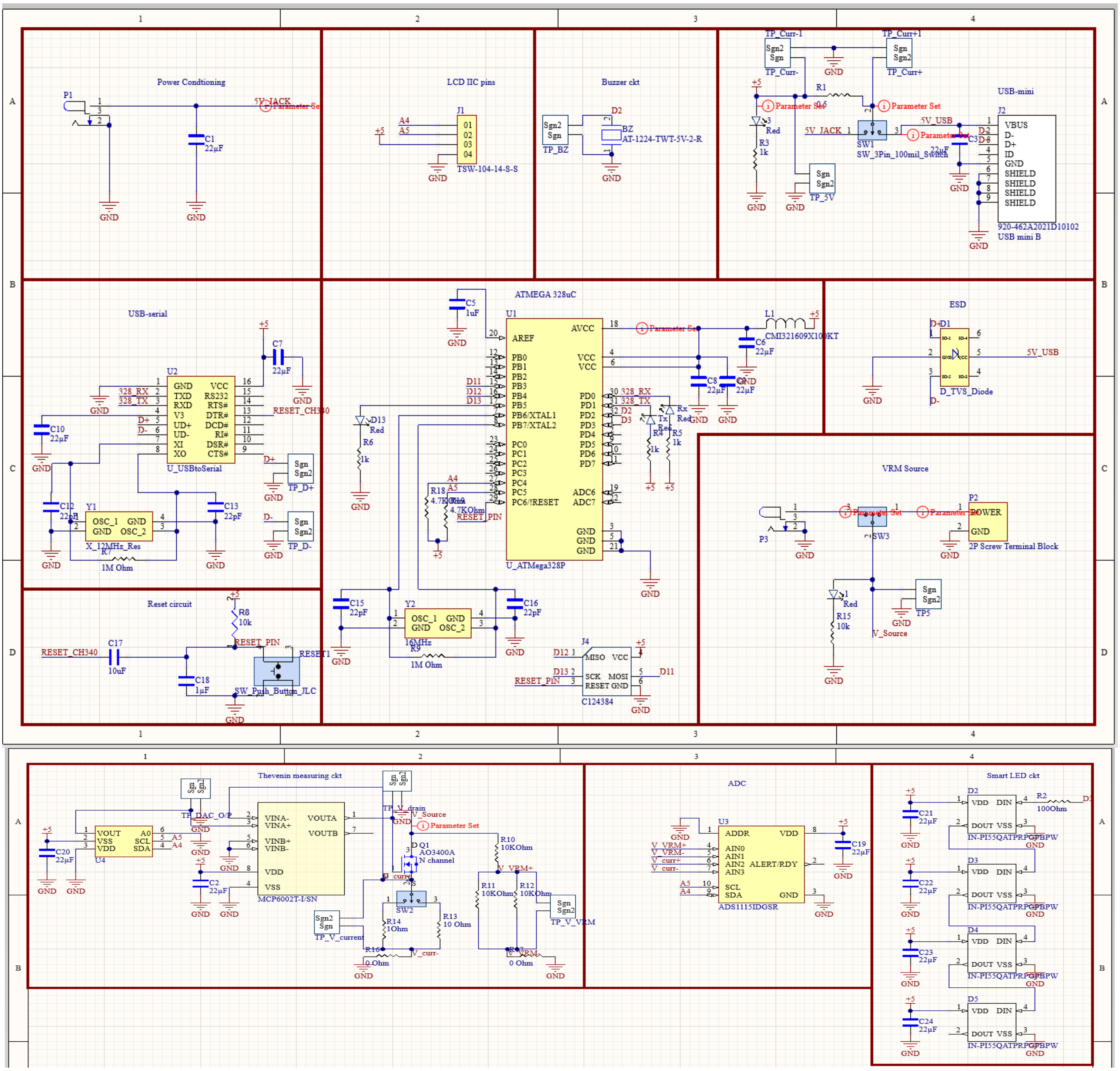
Component listing:

- Microcontroller: Atmega328p
- USB to TTL converter: CH340g
- Crystal oscillator: 12MHz, 16MHz
- Capacitor: 22pF, 1uF, 10uF, 22uF
- Resistor: 500m, 1K, 10K, 1M
- Inductor: 10uH
- Headers
- LEDs: Red
- Switch: push button
- Power jack and USB mini
- DAC: MCP4725
- opAmp: MCP6002
- MOSFET: AO3400A
- ADC: AD1115
- Buzzer: AT-1223TWT-5V
- Smart LEDs: WS2812B-B-W
- Display: HiLetgo 1.3inch OLED

JITHENDRA HS
5/5/24

PCB REPORT – BOARD 4

Schematic:

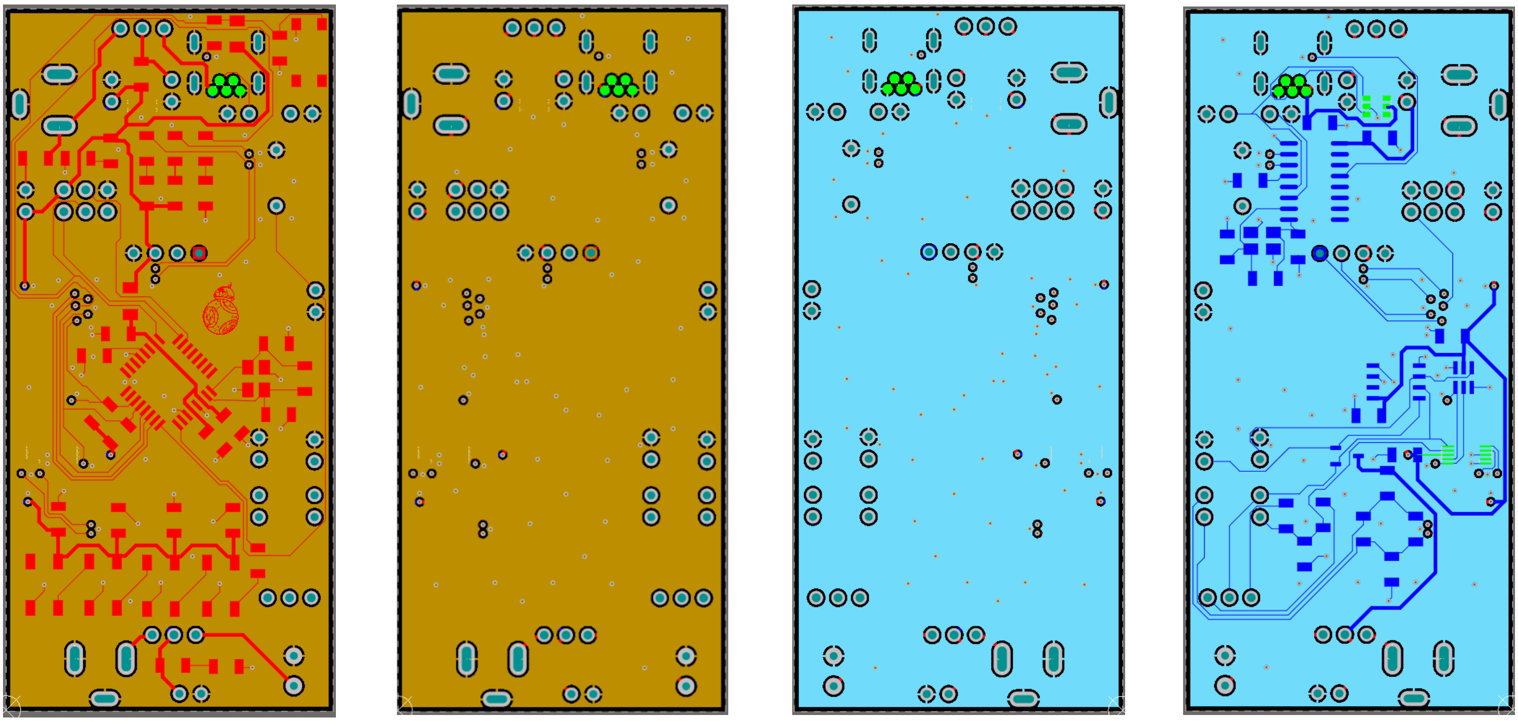


Circuit involved in Instrument droid board.

JITHENDRA HS
5/5/24

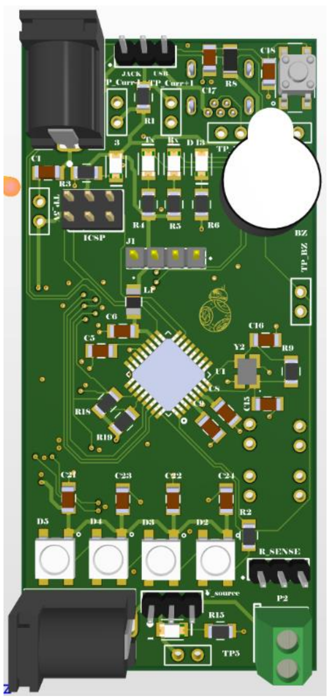
PCB REPORT – BOARD 4

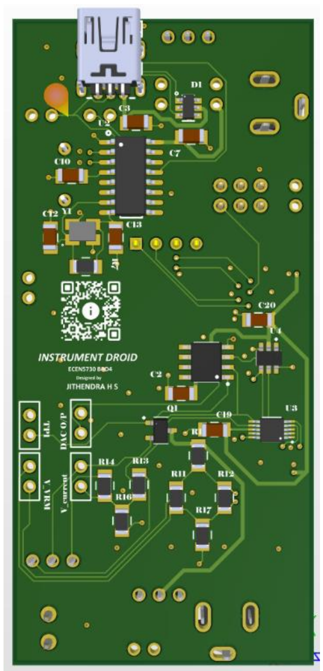
Layout:



4 individual layer placement is Highlighted.

Board:





PCB REPORT – BOARD 4

Assembled board with Display enabled and LED lit up.

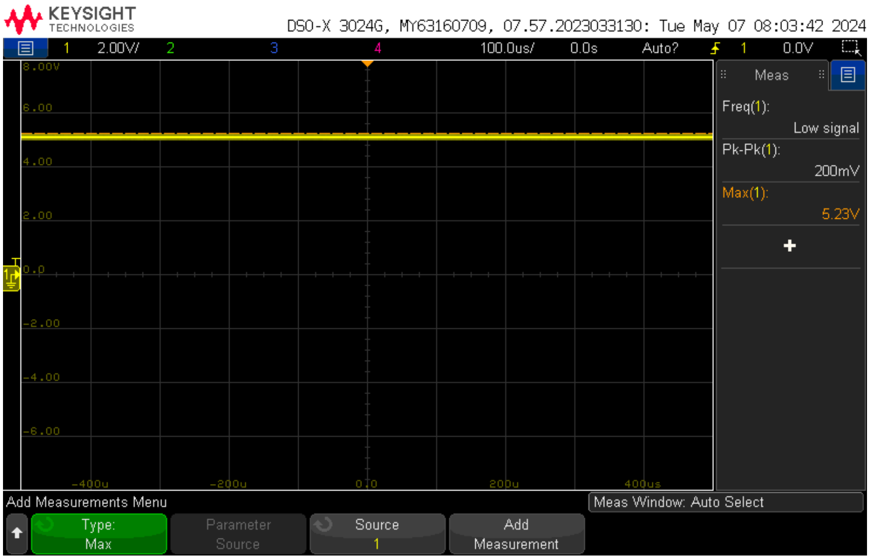
Board bring-up procedure followed:

- Soldered only the required components to bootload the brain-dead ATmega328P microcontroller, such as the microcontroller itself, 16MHz oscillator, ICSP pins, and 5V power supply components.
- Configured a commercial Arduino as an ISP programmer.
- Burned the bootloader code with the help of the dropdown menu under tools in the Arduino IDE.
- Verified the oscillator circuit to ensure it provided a 16MHz frequency.
- Soldered the components required for USB flash.
- Added the DAC, op-amp, MOSFET, and ADC one by one, verifying that each component resulted in the expected waveforms as output.

PCB REPORT – BOARD 4

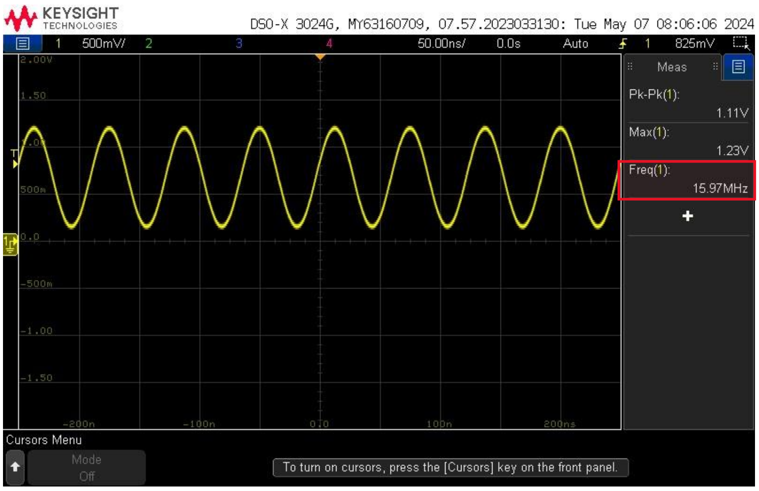
Scope Output:

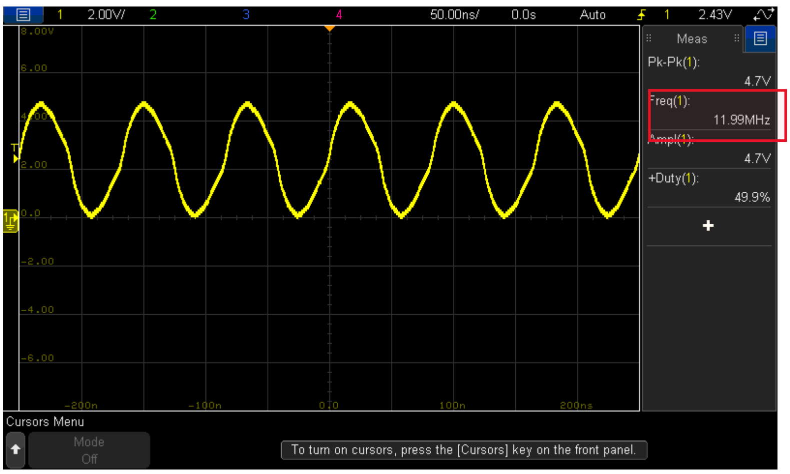
1. **5V power supply:**



Measured and verified that 5V DC voltage is coming from the power jack and USB mini.

2. **Oscillators:**

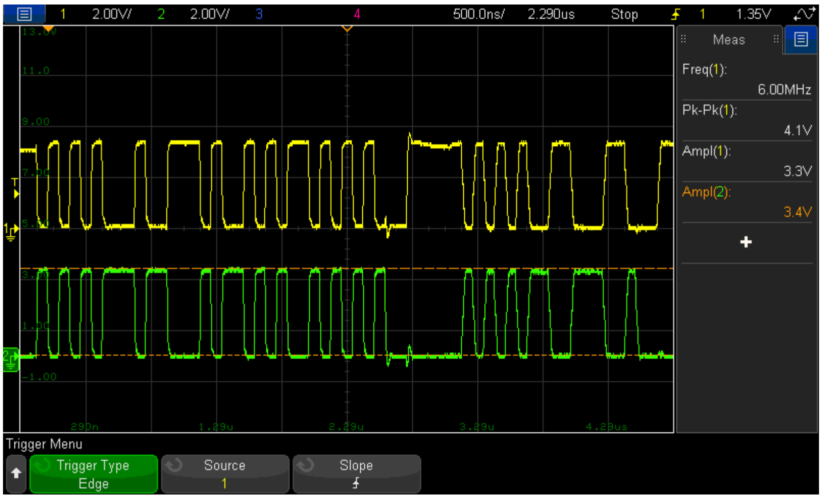




Observed a clock signal of frequency 16MHz generated by the oscillator used for the system clock and 12MHz generated by the oscillator used for USB data transmission.

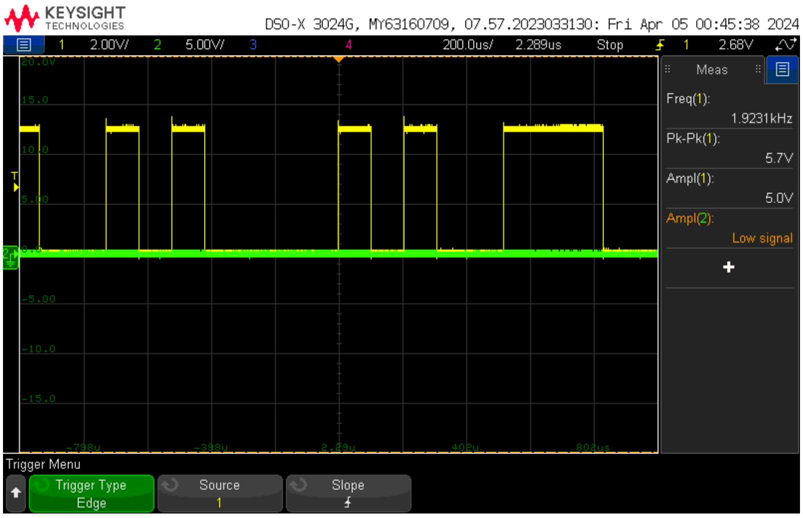
PCB REPORT – BOARD 4

3. **USB mini output:**



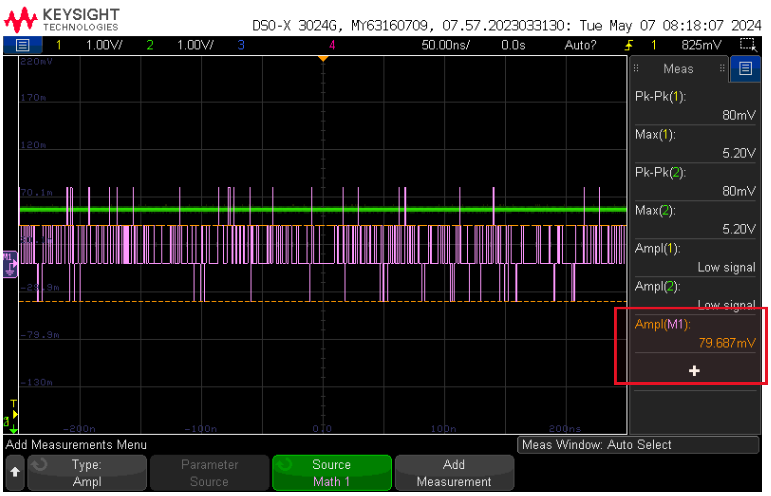
Seen keepalive signal received from D+ and D- of the USB port every 1ms.

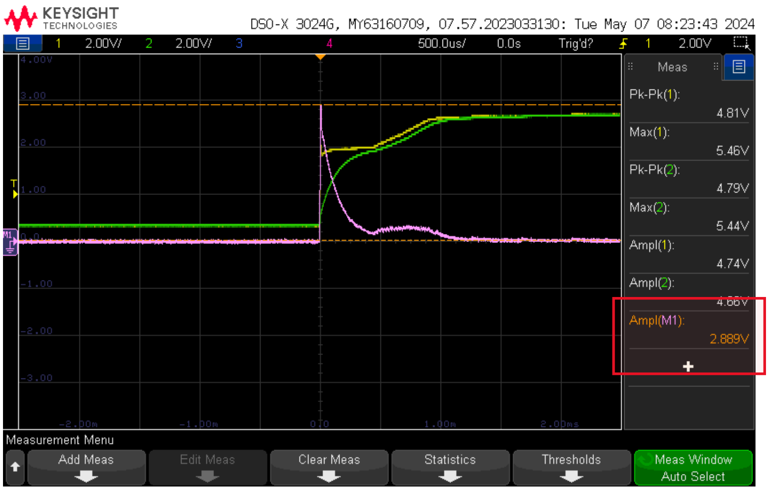
4. **UART output:**



Captured the state of the Tx channel when data was sent out of the Arduino board and confirmed that UART is working fine.

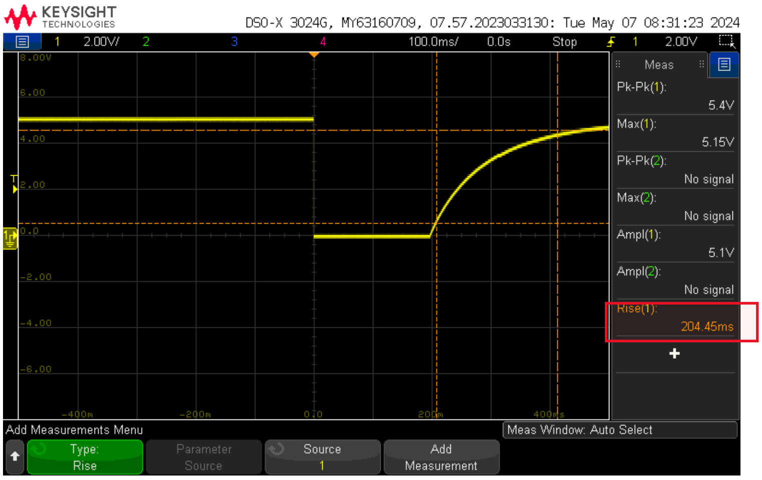
5. **Steady current and In-rush current:**





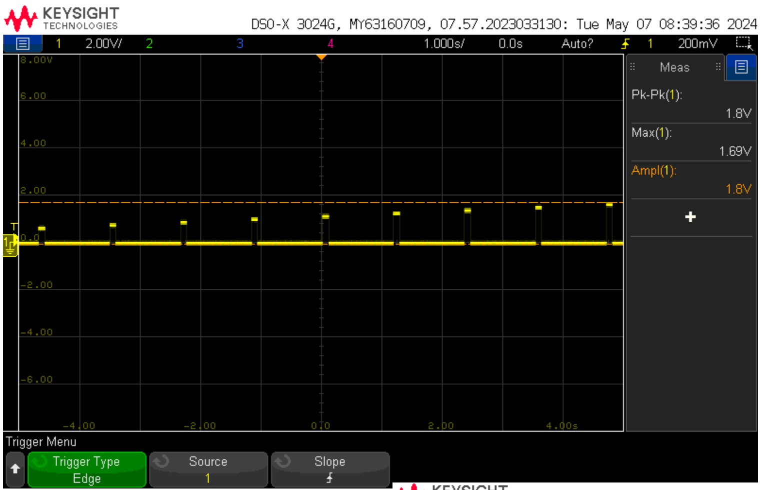
Measured a voltage drop of nearly 80mV across 0.5Ω, indicating a steady-state current draw of about 160mA. Noted a voltage drop of 2.889V upon plugging in the power supply, indicating an inrush current draw of around 5.778A.

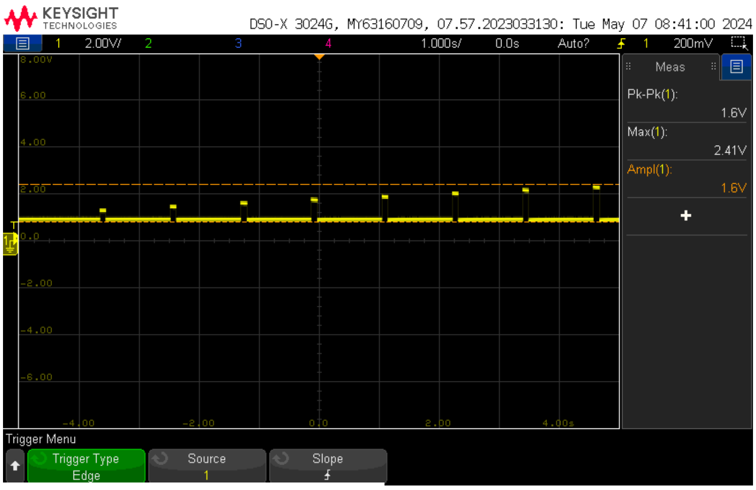
6. **Reset Pin:**

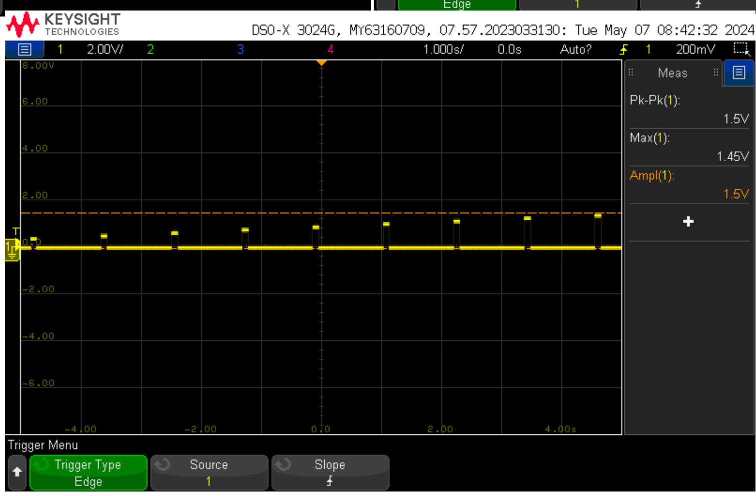


A rise time of 204.45ms was observed at the Golden Arduino board, demonstrating better control over switch debouncing.

7. **DAC, opAmp and MOSFET output:**





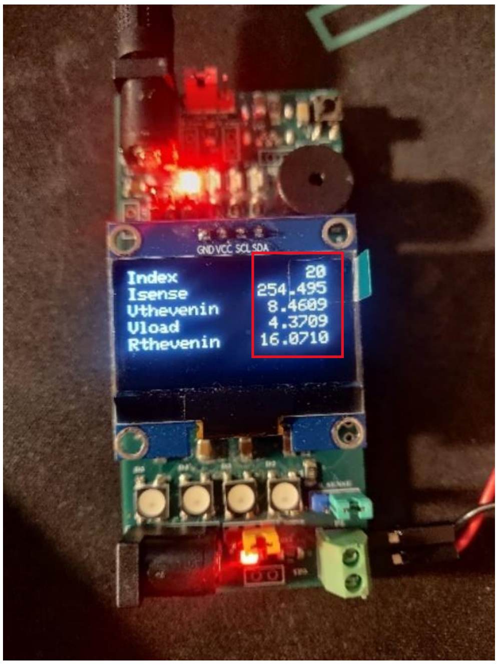


PCB REPORT – BOARD 4

Observed amplitude of voltage output incrementing for every second at DAC and opAmp output to let enough current through MOSFET and sense resistor to find Thevenin resistance.

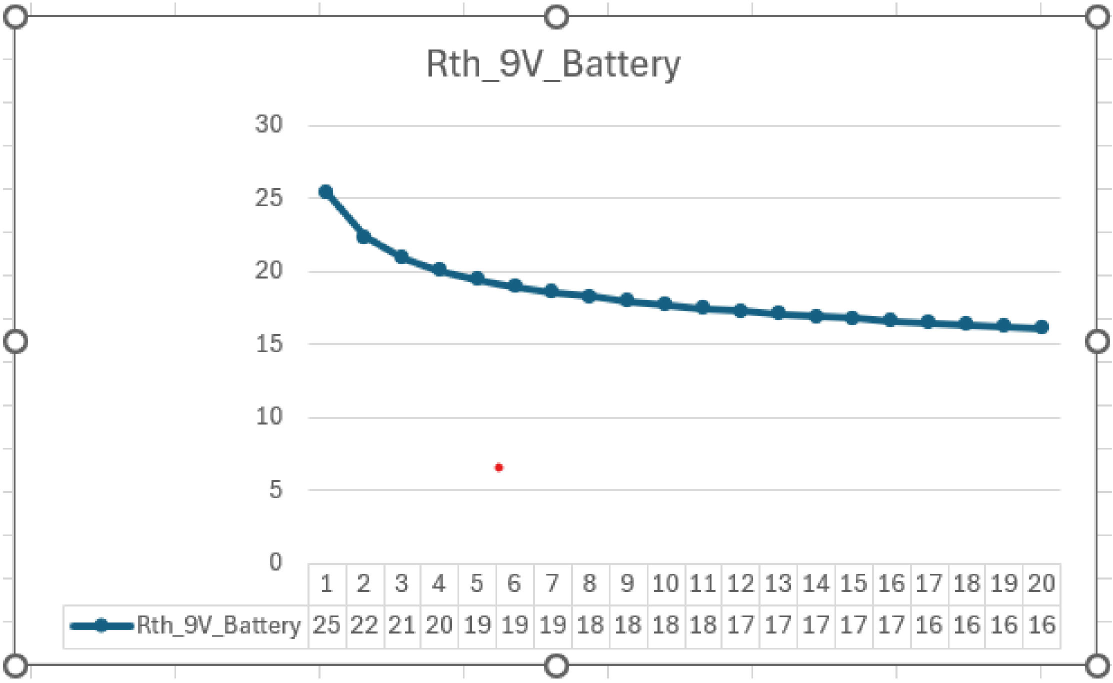
8. Thevenin resistance of 9V battery:

```
20:23:01.947 -> 1, 12.065, 8.5224, 8.2162, 25.3775
20:23:03.084 -> 2, 24.394, 8.5266, 7.9817, 22.3356
20:23:04.165 -> 3, 37.035, 8.5265, 7.7518, 20.9186
20:23:05.298 -> 4, 49.607, 8.5244, 7.5299, 20.0472
20:23:06.399 -> 5, 62.347, 8.5211, 7.3098, 19.4280
20:23:07.488 -> 6, 74.747, 8.5169, 7.0990, 18.9694
20:23:08.606 -> 7, 87.150, 8.5124, 6.8923, 18.5891
20:23:09.715 -> 8, 99.607, 8.5074, 6.6874, 18.2720
20:23:10.805 -> 9, 112.144, 8.5020, 6.4842, 17.9930
20:23:11.948 -> 10, 124.760, 8.4962, 6.2840, 17.7317
20:23:13.055 -> 11, 137.443, 8.4902, 6.0844, 17.5042
20:23:14.150 -> 12, 150.037, 8.4838, 5.8871, 17.3069
20:23:15.249 -> 13, 162.419, 8.4770, 5.6970, 17.1166
20:23:16.350 -> 14, 175.002, 8.4700, 5.5045, 16.9452
20:23:17.458 -> 15, 187.857, 8.4626, 5.3119, 16.7716
20:23:18.558 -> 16, 200.365, 8.4549, 5.1257, 16.6155
20:23:19.699 -> 17, 212.704, 8.4470, 4.9410, 16.4827
20:23:20.793 -> 18, 225.382, 8.4388, 4.7530, 16.3534
20:23:21.885 -> 19, 237.961, 8.4302, 4.5707, 16.2191
20:23:23.022 -> 20, 250.518, 8.4215, 4.3878, 16.1015
20:23:23.022 -> done
```



Reading is captured over both serial plotter and the display attached to the board and noted 16Ω as its Thevenin resistance.

Plot:



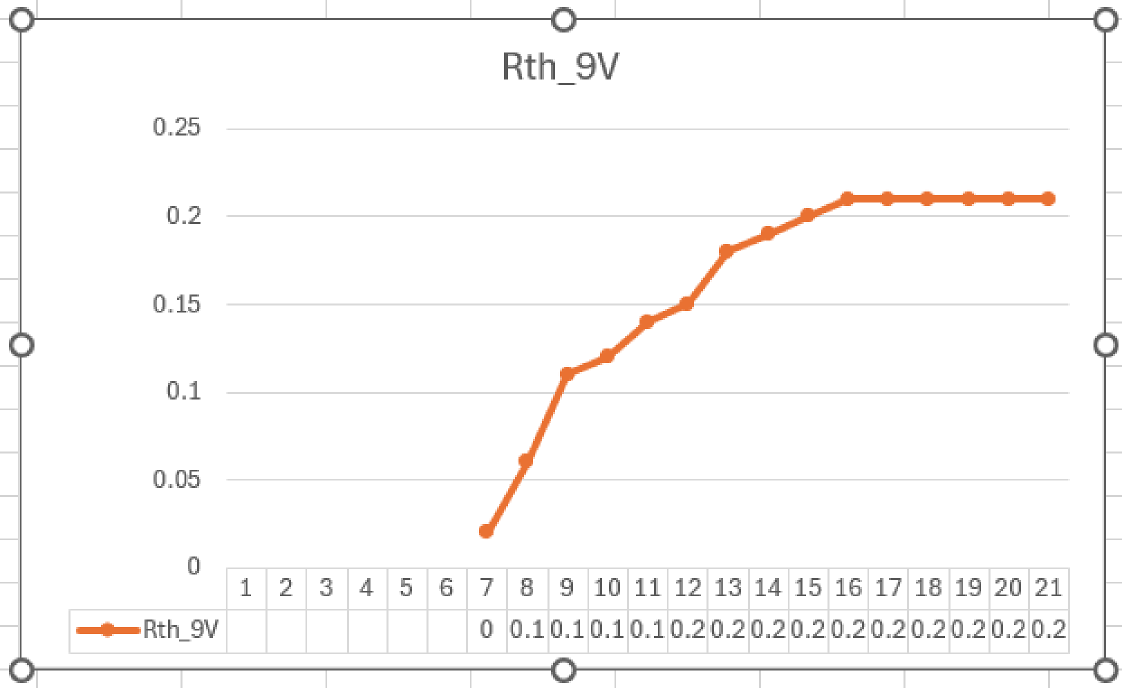
PCB REPORT – BOARD 4

9. Thevenin resistance of 9V adapter:

```
20:36:16.212 -> 1, 12.063, 9.3006, 9.3094, -0.7253
20:36:17.304 -> 2, 24.392, 9.3009, 9.3387, -1.5502
20:36:18.440 -> 3, 37.049, 9.3009, 9.3346, -0.9096
20:36:19.535 -> 4, 49.655, 9.3018, 9.3233, -0.4327
20:36:20.620 -> 5, 62.404, 9.3016, 9.3124, -0.1730
20:36:21.763 -> 6, 74.855, 9.3015, 9.3045, -0.0398
20:36:22.827 -> 7, 87.231, 9.3012, 9.2987, 0.0289
20:36:23.977 -> 8, 99.670, 9.3016, 9.2951, 0.0647
20:36:25.049 -> 9, 112.180, 9.3013, 9.2888, 0.1109
20:36:26.166 -> 10, 124.787, 9.3010, 9.2852, 0.1271
20:36:27.258 -> 11, 137.440, 9.3010, 9.2813, 0.1437
20:36:28.395 -> 12, 150.062, 9.3009, 9.2777, 0.1547
20:36:29.486 -> 13, 162.427, 9.3012, 9.2712, 0.1842
20:36:30.605 -> 14, 175.074, 9.3011, 9.2664, 0.1981
20:36:31.702 -> 15, 187.937, 9.3010, 9.2620, 0.2072
20:36:32.827 -> 16, 200.421, 9.3007, 9.2586, 0.2104
20:36:33.913 -> 17, 212.853, 9.3008, 9.2555, 0.2130
20:36:35.040 -> 18, 225.554, 9.3014, 9.2526, 0.2160
20:36:36.122 -> 19, 238.184, 9.3012, 9.2505, 0.2128
20:36:37.258 -> 20, 250.791, 9.3018, 9.2484, 0.2129
20:36:37.258 -> done
```

Reading is captured over both serial plotter and the display attached to the board and noted 0.21Ω as its Thevenin resistance. Not sure why there is voltage boost in the initial stage!!!

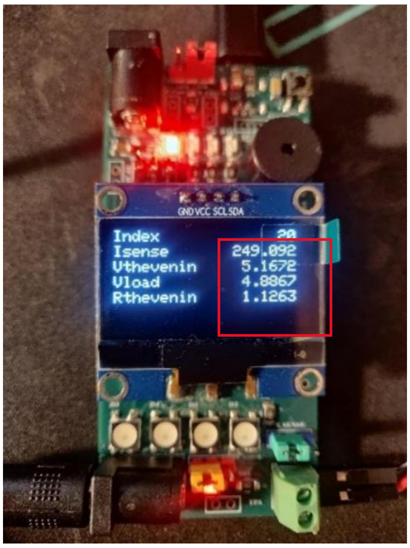
Plot:



PCB REPORT – BOARD 4

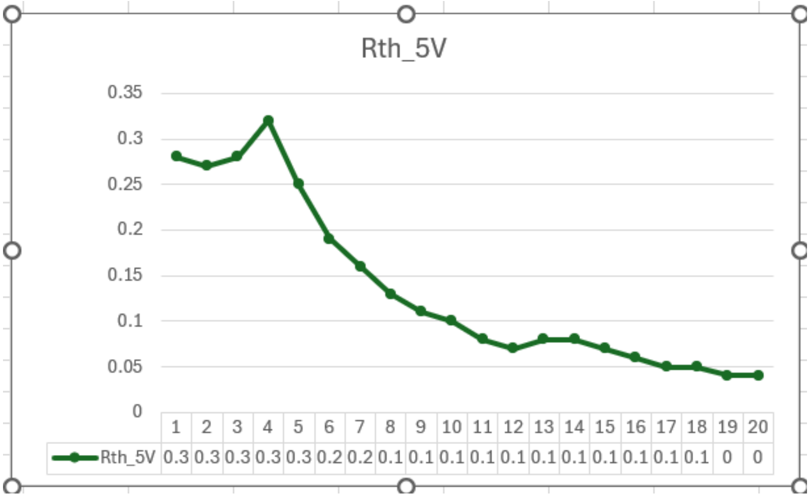
10. Thevenin resistance of 5V adapter:

```
20:28:52.393 -> 1, 10.092, 5.1619, 5.1590, 0.2846
20:28:53.534 -> 2, 21.997, 5.1619, 5.1557, 0.2789
20:28:54.621 -> 3, 33.806, 5.1622, 5.1526, 0.2844
20:28:55.728 -> 4, 45.616, 5.1621, 5.1472, 0.3271
20:28:56.822 -> 5, 58.654, 5.1622, 5.1475, 0.2501
20:28:57.960 -> 6, 70.485, 5.1622, 5.1482, 0.1988
20:28:59.059 -> 7, 82.708, 5.1622, 5.1490, 0.1603
20:29:00.149 -> 8, 94.899, 5.1620, 5.1491, 0.1366
20:29:01.261 -> 9, 108.480, 5.1625, 5.1495, 0.1197
20:29:02.350 -> 10, 120.626, 5.1625, 5.1493, 0.1092
20:29:03.457 -> 11, 132.870, 5.1624, 5.1508, 0.0880
20:29:04.607 -> 12, 145.256, 5.1625, 5.1512, 0.0777
20:29:05.709 -> 13, 158.558, 5.1626, 5.1489, 0.0861
20:29:06.807 -> 14, 170.546, 5.1626, 5.1474, 0.0894
20:29:07.891 -> 15, 182.535, 5.1624, 5.1487, 0.0745
20:29:09.020 -> 16, 194.414, 5.1624, 5.1501, 0.0632
20:29:10.128 -> 17, 207.564, 5.1627, 5.1510, 0.0560
20:29:11.228 -> 18, 219.576, 5.1627, 5.1514, 0.0515
20:29:12.331 -> 19, 231.666, 5.1627, 5.1521, 0.0458
20:29:13.445 -> 20, 243.993, 5.1630, 5.1525, 0.0431
20:29:13.445 -> done
```

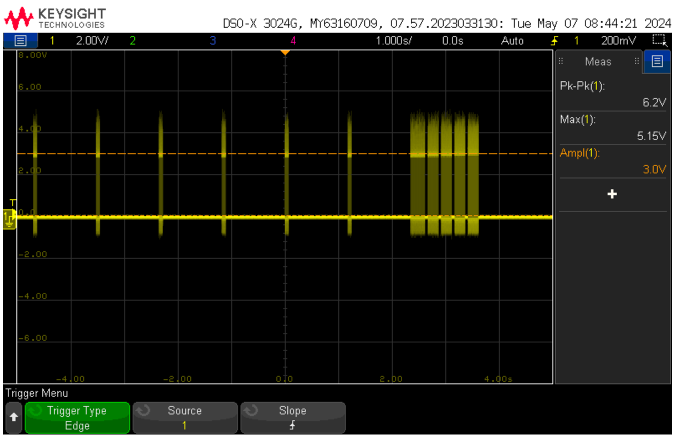


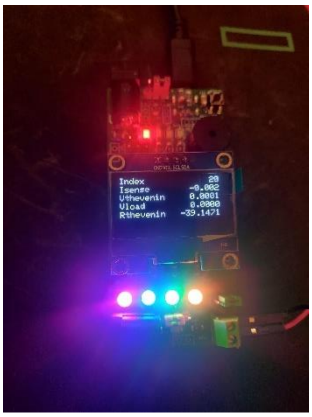
Reading is captured over both serial plotter and the display attached to the board and noted 0.04Ω as and 1.1263 as its Thevenin resistance. Not sure of why there is difference!!!

Plot:



11. Buzzer and Smart Led:





Pulse sent towards buzzer and Smart LED's toggle are captured.

PCB REPORT – BOARD 4

12. What all worked

characteristics	Result	Remarks
Obtaining a 5V power supply from the power jack and USB mini.	Worked	
Utilizing a 12MHz frequency oscillator for the USB data transmission.	Worked	
Utilizing a 16MHz oscillator frequency for the system clock.	Worked	
Data transmission via D+ and D- pins.	Worked	
Data transmission via Tx and Rx.	Worked	
Implementing proper microcontroller reset without debouncing.	Worked	
Ensuring that LED indicators are working fine.	Worked	
No signal integrity problem wrt to Via's.	Worked	
Able to connect the voltage source through power jack and Screw terminal.		
Able to communicate and modify DAC output.	Worked	
Able to communicate and read ADC output.	Worked	
Able to measure the Thevenin Voltage and Thevenin resistance.	Worked	
Able to display the reading in real time.		

Debug:

- I2C communication was not working due to interchange VSS and VDD pins connected to DAC. After finding the problem rewired it externally.

Mistakes:

- Missed to label few test points.
- D+, D-, RX and TX Signal trace should have been considered as differential pairs, so should have same trace length.

Key learnings

- 4 Layer development best practices such adding Via's near to signal transition from one layer to another to accommodate shortest path for return current, which also plays major role in reducing inductive noise in multilayer boards.
- Principle of Voltage source characterizer.
- How to use Arduino with ADC to read and compare differential voltage.
- How to bring up the board starting from boot loading the Arduino till interfacing DAC, opAmp, MOSFET and ADC in incremental order.
- Behaviour of voltage source stable output with respect to current draw from the load.
- Limitation of Voltage source by drawing more current which can drop voltage up to 75% of ratings.

Appendix:

```
#include <SPI.h>
#include <Wire.h>
#include <Adafruit_GFX.h>
#include <Adafruit_SH110X.h>
#include <Adafruit_MCP4725.h>
#include <Adafruit_ADS1X15.h>

#define i2c_Address 0x3c
#define SCREEN_WIDTH 128 // OLED display width, in pixels
#define SCREEN_HEIGHT 64 // OLED display height, in pixels
#define OLED_RESET -1    // QT-PY / XIAO
Adafruit_SH1106G display = Adafruit_SH1106G(SCREEN_WIDTH, SCREEN_HEIGHT,
&Wire, OLED_RESET);

#define BUZZER_PIN 2

Adafruit_ADS1115 ads;
Adafruit_MCP4725 dac;
float R_sense = 10; //current sensor
long itime_on_msec = 100; //on time for taking measurements
long itime_off_msec = itime_on_msec * 10; // time to cool off
int iCounter_off = 0; // counter for number of samples off
int iCounter_on = 0; // counter for number of samples on
float v_divider = 5000.0 / 15000.0; // voltage divider on the VRM
float DAC_ADU_per_v = 4095.0 / 5.0; //conversion from volts to ADU
int V_DAC_ADU; // the value in ADU to output on the DAC
int I_DAC_ADU; // the current we want to output
float I_A = 0.0; //the current we want to output, in amps
long itime_stop_usec; // this is the stop time for each loop
float ADC_V_per_ADU = 0.125 * 1e-3; // the voltage of one bit on the gain of 1
float V_VRM_on_v; // the value of the VRM voltage
float V_VRM_off_v; // the value of the VRM voltage
float I_sense_on_A; // the current through the sense resistor
float I_sense_off_A; // the current through the sense resistor
float I_max_A = 0.25; // max current to set for
int npts = 20; //number of points to measure
float I_step_A = I_max_A / npts; //step current change
float I_load_A; // the measured current load
float V_VRM_thevenin_v;
float V_VRM_loaded_v;
float R_thevenin;
int i;
void setup() {
  pinMode(BUZZER_PIN, OUTPUT);
```

```
  delay(250); // wait for the OLED to power up
  display.begin(i2c_Address, true); // Address 0x3C default
  display.clearDisplay();
  testdrawroundrect();
  delay(1000);
  display.clearDisplay();

  display.setTextSize(2);
  display.setTextColor(SH110X_WHITE);
  display.setCursor(0, 18);
  display.println("Instrument");
  display.setCursor(30,38);
  display.println("Droid");
  display.display();
  delay(2000);
  display.clearDisplay();


  dac.begin(0x60); // address is either 0x60, 0x61, 0x62,0x63, 0x64 or 0x65
  dac.setVoltage(0, false); //sets the output current to 0 initially

  // ads.setGain(GAIN_TWOTHIRDS); // 2/3x gain +/- 6.144V 1 bit = 3mV 0.1875mV
(default)
  ads.setGain(GAIN_ONE); // 1x gain +/- 4.096V 1 bit = 2mV 0.125mV
  // ads.setGain(GAIN_TWO); // 2x gain +/- 2.048V 1 bit = 1mV 0.0625mV
  // ads.setGain(GAIN_FOUR); // 4x gain +/- 1.024V 1 bit = 0.5mV 0.03125mV
  // ads.setGain(GAIN_EIGHT); // 8x gain +/- 0.512V 1 bit = 0.25mV 0.015625mV
  // ads.setGain(GAIN_SIXTEEN); // 16x gain +/- 0.256V 1 bit = 0.125mV
0.0078125mV
  ads.begin(0x48); // note- you can put the address of the ADS111 here if
needed
  ads.setDataRate(RATE_ADS1115_860SPS);// sets the ADS1115 for higher speed

  //Serial.println("Setup done");
}

void loop() {
  noTone(BUZZER_PIN);
  for (i = 1; i <= npts; i++) {
    display.clearDisplay();
    I_A = i * I_step_A;
    dac.setVoltage(0, false); //sets the output current
    func_meas_off();

    func_meas_on();
    dac.setVoltage(0, false); //sets the output current
    I_load_A = I_sense_on_A - I_sense_off_A; //load current
```

```
    V_VRM_thevenin_v = V_VRM_off_v;
    V_VRM_loaded_v = V_VRM_on_v;
    R_thevenin = (V_VRM_thevenin_v - V_VRM_loaded_v) / I_load_A;
    if (V_VRM_loaded_v < 0.25 * V_VRM_thevenin_v)
    i = npts; //stops the ramping

  tone(BUZZER_PIN, 1000);

  display.setTextSize(1);
  char indexformattedString[32];
  int index = i;
  char intString[3];
  itoa(index, intString, 10);
  sprintf(indexformattedString, "Index             %s", intString);
  display.setCursor(0,0);
  display.println(indexformattedString);

  char senseCurrentformattedString[32];
  float senseCurrent = I_load_A*1000;
  char senseCurrentString[10];
  dtostrf(senseCurrent, 10, 3, senseCurrentString);
  display.setCursor(0,10);
  sprintf(senseCurrentformattedString, "Isense    %s", senseCurrentString);
  display.println(senseCurrentformattedString);

  char voltageTheveninformattedString[32];
  float voltageThevenin = V_VRM_thevenin_v;
  char voltageTheveninString[10];
  dtostrf(voltageThevenin, 10, 4, voltageTheveninString);
  display.setCursor(0,20);
  sprintf(voltageTheveninformattedString, "Vthevenin %s",
voltageTheveninString);
  display.println(voltageTheveninformattedString);

  char voltageLoadformattedString[32];
  float voltageLoad = V_VRM_loaded_v;
  char voltageLoadString[10];
  dtostrf(voltageLoad, 10, 4, voltageLoadString);
  display.setCursor(0,30);
  sprintf(voltageLoadformattedString, "Vload     %s", voltageLoadString);
  display.println(voltageLoadformattedString);

  char resistanceTheveninformattedString[32];
  float resistanceThevenin = R_thevenin;
  char resistanceTheveninString[10];
  dtostrf(resistanceThevenin, 10, 4, resistanceTheveninString);
  display.setCursor(0,40);
```

```
  sprintf(resistanceTheveninformattedString, "Rthevenin %s",
resistanceTheveninString);
  display.println(resistanceTheveninformattedString);
  display.display();
  noTone(BUZZER_PIN);
  }
  // Play the tune for "task completed"
  tone(BUZZER_PIN, 1000, 200); // C
  delay(250);
  tone(BUZZER_PIN, 1200, 200); // D
  delay(250);
  tone(BUZZER_PIN, 1400, 200); // E
  delay(250);
  tone(BUZZER_PIN, 1600, 200); // F
  delay(250);
  tone(BUZZER_PIN, 1800, 200); // G
  delay(250);
  noTone(BUZZER_PIN); // Silence
  delay(10000);

}

void testdrawroundrect(void) {
  for (int16_t i = 0; i < display.height() / 2 - 2; i += 2) {
    display.drawRoundRect(i, i, display.width() - 2 * i, display.height() - 2
* i, display.height() / 4, SH110X_WHITE);
    display.display();
    delay(1);
  }
}

void func_meas_off() {
  dac.setVoltage(0, false); //sets the output current
  iCounter_off = 0; //starting the current counter
  V_VRM_off_v = 0.0; //initialize the VRM voltage averager
  I_sense_off_A = 0.0; // initialize the current averager
  itime_stop_usec = micros() + itime_off_msec * 1000; // stop time
  while (micros() <= itime_stop_usec) {
    V_VRM_off_v = ads.readADC_Differential_0_1() * ADC_V_per_ADU / v_divider +
                  V_VRM_off_v;
    I_sense_off_A = ads.readADC_Differential_2_3() * ADC_V_per_ADU / R_sense +
                    I_sense_off_A;
    iCounter_off++;
  }
  V_VRM_off_v = V_VRM_off_v / iCounter_off;
  I_sense_off_A = I_sense_off_A / iCounter_off;
  // Serial.print(iCounter_off);Serial.print(", ");
```

```
  // Serial.print(I_sense_off_A * 1e3, 4); Serial.print(", ");
  // Serial.println(V_VRM_off_v, 4);
}

void func_meas_on() {
  //now turn on the current
  I_DAC_ADU = I_A * R_sense * DAC_ADU_per_v;
  dac.setVoltage(I_DAC_ADU, false); //sets the output current
  iCounter_on = 0;
  V_VRM_on_v = 0.0; //initialize the VRM voltage averager
  I_sense_on_A = 0.00; // initialize the current averager
  itime_stop_usec = micros() + itime_on_msec * 1000; // stop time
  while (micros() <= itime_stop_usec) {
    V_VRM_on_v = ads.readADC_Differential_0_1() * ADC_V_per_ADU / v_divider +
                 V_VRM_on_v;
    I_sense_on_A = ads.readADC_Differential_2_3() * ADC_V_per_ADU / R_sense +
                   I_sense_on_A;
    iCounter_on++;
  }
  dac.setVoltage(0, false); //sets the output current to zero
  V_VRM_on_v = V_VRM_on_v / iCounter_on;
  I_sense_on_A = I_sense_on_A / iCounter_on;
  // Serial.print(iCounter_on);Serial.print(", ");
  // Serial.print(I_sense_on_A * 1e3, 4);Serial.print(", ");
  // Serial.println(V_VRM_on_v, 4);
}
```