Utkarsh Mandavilli
ECEN 5730
4/27/26
Final Report

Arduino and Power Supply Test Droid

Project Overview:

The purpose of this board was to gain practice designing with 4-layers. It contains an arduino circuit, the same found in Board 3, but also has a VRM Test circuit and peripherals in the form of Smart LEDs and a buzzer. When approaching the design for this board, I utilized the top layer for the majority of the components and the main ICs. The bottom layer was used for any components that needed to be close to the microcontroller but would complicate routing if placed on top. It also contained the Smart LEDs due to their relatively low profile and large footprint. The separate sections minimized trace lengths and prevented the use of cross-unders. This allowed us to maintain a continuous ground plane.

Several design issues identified in the previous iteration of Board 3, were addressed in this version. We were able to decrease the number of header pins due to this being a standalone board. We also ensured that the clock circuits required for the ATMega 328 and the CH340G had the capacitors placed correctly. The layout also ensured that there was enough space between ICs and nearby components to allow for soldering to be achieved easily without damaging parts. The layout could still be improved upon however. Planning component placement ahead of time will ensure an even more simple routing and assembly process.

POR:

1. The board shall be powered by a 5V AC/DC power regulator or through a micro-USB connection.
2. An indicator LED shall signal when the board is powered.
3. Additional indicator LEDs shall signal important components being powered.
4. The Atmega 328 shall be bootloaded to become a functional Arduino board.
5. The Arduino IDE should be used to run a sketch on the board.
6. A piezoelectric buzzer shall be included and operable through one of the ATmega's programmable pins.
7. At least 4 smart RGB LEDs shall be included and operable through a programmable ATmega pin
8. The VRM shall be able to receive an input through a power jack, micro-USB port, or a screw connector.

9. The board shall create an electronic load to draw current from a voltage source and measure the resulting voltage drop.
10. Data acquired from the VRM shall be viewable through an Arduino Serial Monitor and/or Serial Plotter.

Bill of Materials:

Name	Designator	Quantity
Buzzer	BZ1	1
22pF	C1, C2, C3, C4	4
1uF	C5, C11	2
22uF	C6, C7, C8, C9, C10, C12, C13, C14	8
10x Probe TP	Current 1, Current 2, RX1, TX1	4
SRV05-4-P-T7	D1	1
Smart LEDs	D2, D3, D4, y4	4
USB Mini 2	J1	1
USB Mini 1	J2	1
NRPN032PAEN-RC	J3	1
10uH	L1	1
Red LEDs	LED1, LED2, LED3, LED4, LED5	5
Power Jack	P1, P2	2
2P Screw Terminal Block	P3	1
0 Ohm	R1, R14	2
1M	R2, R3	2
47 Ohms	R4, R8, R9, R13, R15	5
10k	R5, R6, R7, R16, R17, R18	6
10 Ohm	R10	1
1 Ohm	R11	1
500m	R12	1

100Ohm	R19, R20	2
2 Pin Headers	SW1, SW2, SW3, SW4	4
Button	SW5	1
ADS1115IDGSR	U1	1
MCP6002T-I/SN	U2	1
DAC IC	U3	1
ATMEGA328P-ANR	U4	1
CH340G	U5	1
Temp Sensor	U6	1
AO3400A	y3	1
12MHz	Y1	1
16MHz	Y2	1

Schematic:

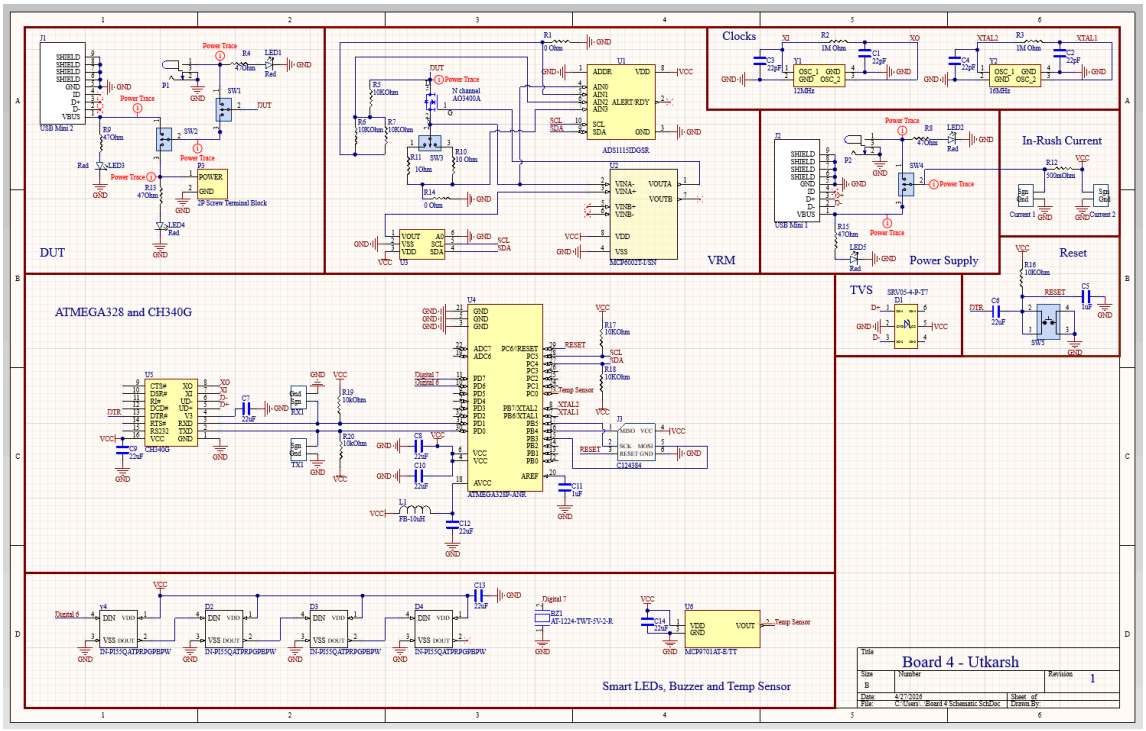


Figure 1: Schematic in Altium Designer.

Final Layout:

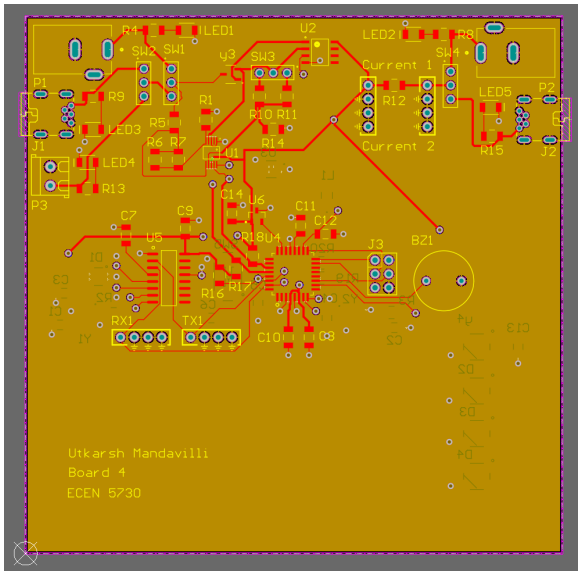


Figure 2: Front of Board Layout in Altium Designer.

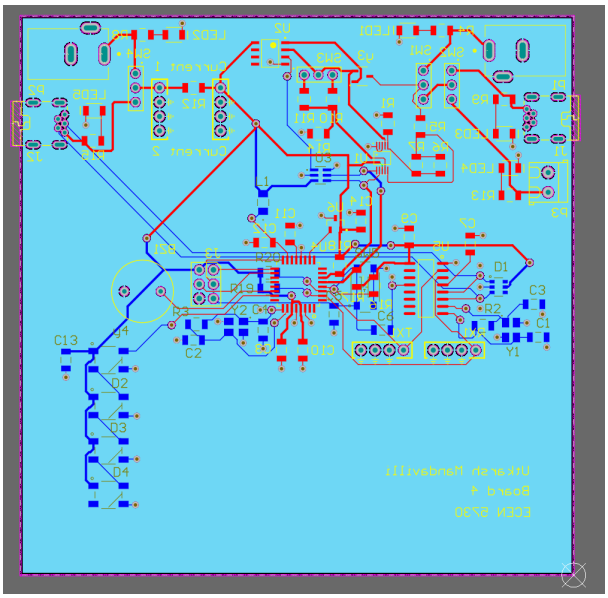


Figure 3: Back of Board Layout in Altium Designer.

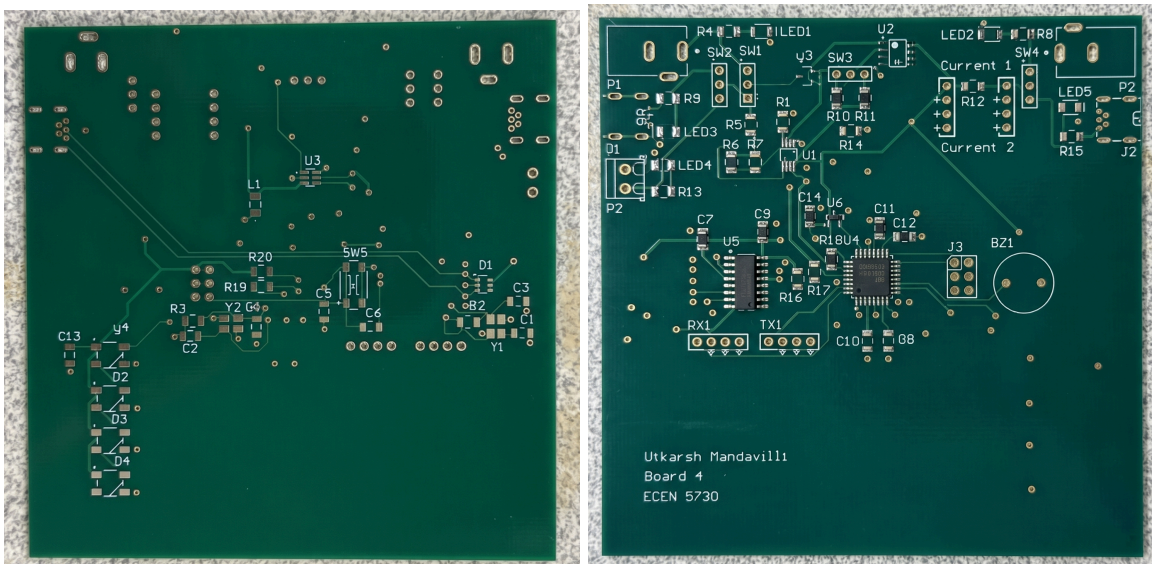


Figure 4: Final Unsoldered Board.

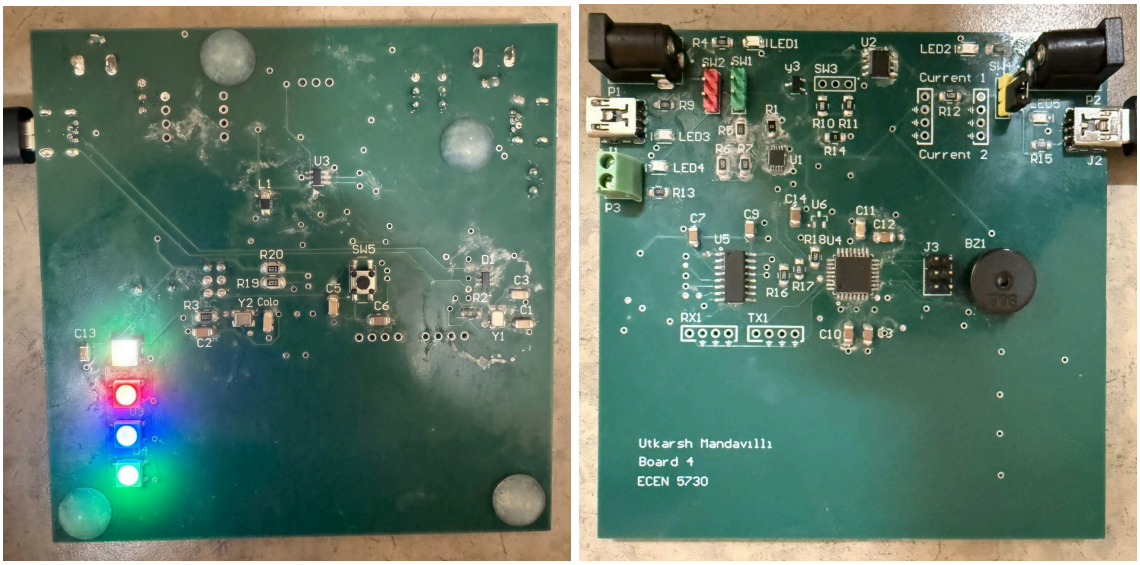


Figure 5: Final Soldered Board.

Here we can see that the smart LEDs work as expected. They turn on once the final measurement of the VRM Test circuit is captured. The buzzer was able to play a song that was converted into a series of tones using a website. This website allowed us to download a file that could be uploaded as arduino code.

VRM Test Circuit:

The VRM test circuit is designed to measure the power supply, Thevenin voltage, and resistance profiles across varying output load currents. This circuit consists of two primary stages: a voltage divider and an op-amp regulator. The voltage divider is utilized to measure the V_thevenin and enables the use of high-voltage sources. The op-amp is supplied with a DAC to set a specific voltage, which activates a MOSFET to draw a known current through the DUT. This allows for the calculation of R_thevenin by comparing V_VRM and V_current and dividing the result by the R_sense current. Furthermore, the inclusion of an ADC allows the Arduino to acquire and transmit the results to the serial monitor.

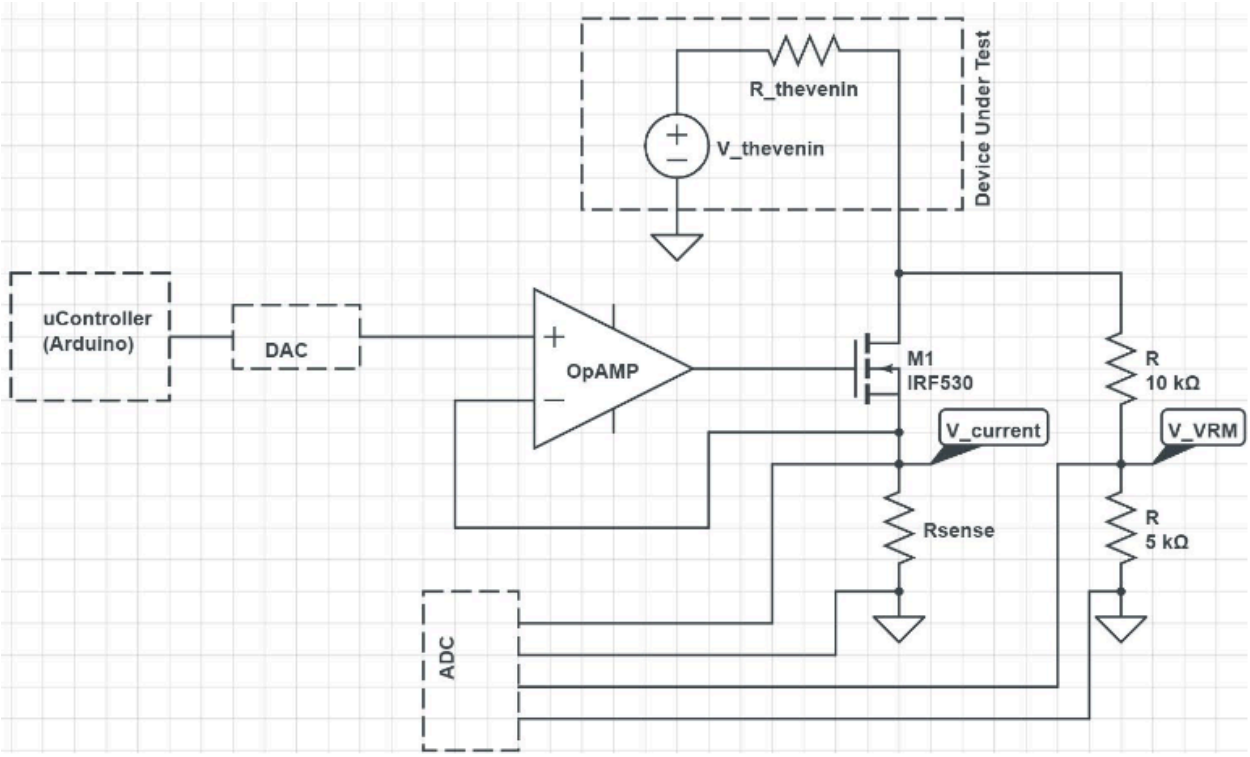
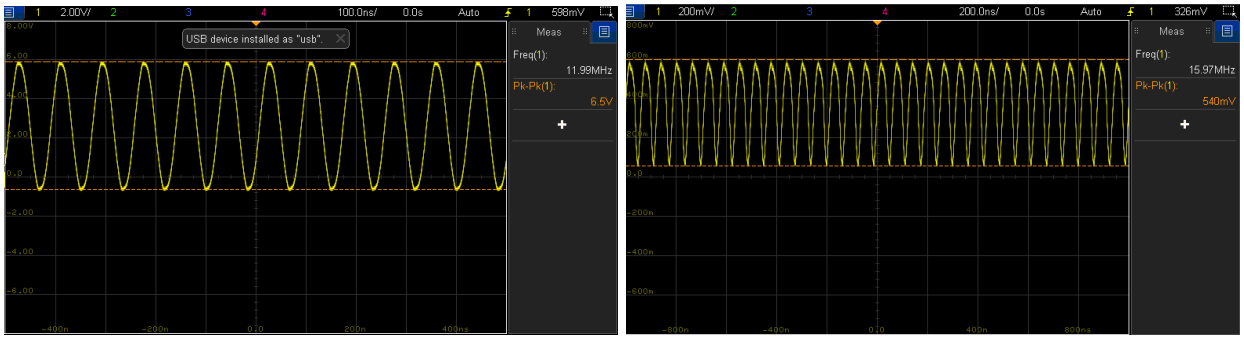


Figure 6: Schematic of VRM Test Circuit.

Results:



Figures 7&8: Scope Measurements of the 12 and 16 MHz clocks respectively.

The frequency for both clocks is as expected. The waveforms are clean and lack noise.

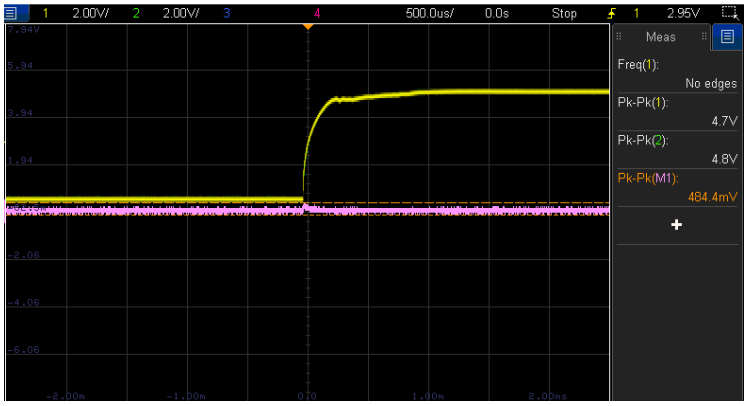


Figure 9: Green is measurement taken at Current 1 and yellow is taken at Current 2. A math function is used to subtract the two channels and is shown in pink.

This was used to calculate the inrush current of the circuit when turned on. Since we used a 100m Ohm resistor and we have a voltage drop of 484.4 mV, we get an inrush current of 4.84A. For all of the testing we use the following code snippet.

```
void setup() {    DDRB
= B00111111;
pinMode(7, OUTPUT);
digitalWrite(7, LOW);
}
void loop() {    PORTB
= B00111101;
delayMicroseconds(4);
PORTB = B00000001;
delay(1);
   digitalWrite(7,
HIGH);
delayMicroseconds(400);
digitalWrite(7, LOW);
delay(10);
}
```

Figure 10: Arduino Code to create a switching transient load.

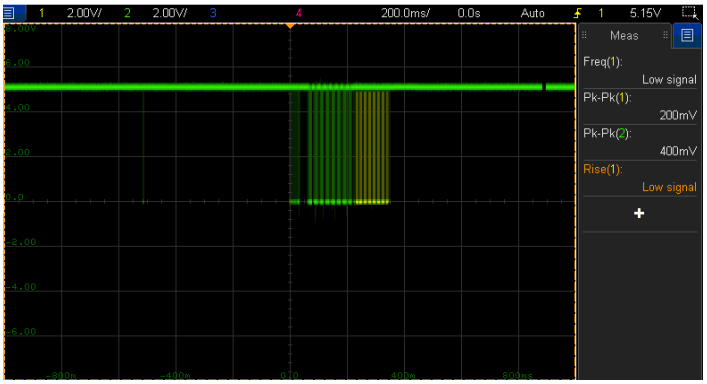


Figure 11: RX line shown in green and TX line shown in yellow.

This figure shows that the ATMEGA is receiving a signal on the RX line from the CH340 and then proceeds to send information back on the TX line.

The VRM Test circuit also behaved as expected when measuring the output from an Agilent 33521A Waveform Generator. In a previous lab we determined an R_thevenin of approximately 50 Ohms and a V_thevenin of approximately 5 Ohms for all loads. This matches the output detected by our board in the serial monitor as shown below.

-> 1, 7.942, 4.5881, 4.2152, 45.4112
-> 2, 20.855, 4.8219, 3.8375, 47.5188
-> 3, 34.082, 4.9402, 3.3341, 48.6521
-> 4, 45.471, 4.7881, 2.6405, 47.1583
-> 5, 59.043, 4.7852, 2.0034, 47.1795
-> 6, 71.412, 4.7877, 1.4012, 47.3781
-> 7, 83.755, 4.7850, 0.7941, 47.6862
-> 8, 96.141, 4.7871, 0.1822, 47.9099

-> 9, 97.665, 4.7858, 0.1045, 47.9481
-> 10, 97.642, 4.7872, 0.1031, 47.9568
-> 11, 97.681, 4.7859, 0.1042, 47.9455
-> 12, 97.668, 4.7871, 0.1034, 47.9392
-> 13, 97.652, 4.7853, 0.1041, 47.9588
-> 14, 97.659, 4.7873, 0.1035, 47.9411
-> 15, 97.674, 4.7857, 0.1043, 47.9482
-> 16, 97.681, 4.7872, 0.1034, 47.9351
-> 17, 97.625, 4.7855, 0.1042, 47.9682
-> 18, 97.663, 4.7871, 0.1032, 47.9463
-> 19, 97.679, 4.7854, 0.1043, 47.9415
-> 20, 97.691, 4.7876, 0.1033, 47.9312

Figure 12: Serial Output in Arduino IDE of VRM Test Circuit.

Conclusion:

Based on our findings we can see that our board performed quite well despite previous challenges encountered with design. The majority of the items in the POR were achieved and all aspects of the board were functional. I gained quite a lot of practice with routing on a 4-layer board while still maintaining best design practices. We can also see the benefits of a VRM Test Circuit in measuring power supplies.

Code:

```
#include <Wire.h>
#include <Adafruit_MCP4725.h>
#include <Adafruit_ADS1X15.h>
#include <Adafruit_NeoPixel.h>

// --- Hardware Configuration ---
#define BUZZER_PIN 7
#define LED_PIN    6
#define LED_COUNT  4
#define BRIGHTNESS 50

// --- Library Objects ---
Adafruit_ADS1115 ads;
Adafruit_MCP4725 dac;
Adafruit_NeoPixel strip(LED_COUNT, LED_PIN, NEO_GRBW + NEO_KHZ800);

// --- VRM Characterizer Variables ---
float R_sense = 4.7;
long itime_on_msec = 100;
long itime_off_msec = itime_on_msec * 10;
int iCounter_off = 0;
int iCounter_on = 0;
float v_divider = 5000.0 / 15000.0;
float DAC_ADU_per_v = 4095.0 / 5.0;
int V_DAC_ADU;
int I_DAC_ADU;
float I_A = 0.0;
long itime_stop_usec;
float ADC_V_per_ADU = 0.125 * 1e-3;
float V_VRM_on_v;
float V_VRM_off_v;
float I_sense_on_A;
float I_sense_off_A;
float I_max_A = 0.25;
int npts = 20;
float I_step_A = I_max_A / npts;
float I_load_A;
float V_VRM_thevenin_v;
float V_VRM_loaded_v;
float R_thevenin;
int i;

int tones[] = {200, 300, 400, 500, 600};
```

```
void setup() {
  Serial.begin(115200);

  strip.begin();
  strip.show();
  strip.setBrightness(BRIGHTNESS);

  dac.begin(0x60);
  dac.setVoltage(0, false);

  ads.setGain(GAIN_ONE);
  ads.begin(0x48);
  ads.setDataRate(RATE_ADS1115_860SPS);
}

void loop() {
  strip.clear();
  strip.show();
  for (i = 1; i <= npts; i++) {
    I_A = i * I_step_A;
    dac.setVoltage(0, false);
    func_meas_off();
    func_meas_on();
    dac.setVoltage(0, false);

    I_load_A = I_sense_on_A - I_sense_off_A;
    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 < 0.75 * V_VRM_thevenin_v) i = npts;

    Serial.print(i);
    Serial.print(", ");
    Serial.print(I_load_A * 1e3, 3);
    Serial.print(", ");
    Serial.print(V_VRM_thevenin_v, 4);
    Serial.print(", ");
    Serial.print(V_VRM_loaded_v, 4);
    Serial.print(", ");
    Serial.println(R_thevenin, 4);
  }

  Serial.println("done");

  strip.setPixelColor(0, strip.Color(0, 0, 0, 255)); // Pixel 1: White
  strip.setPixelColor(1, strip.Color(255, 0, 0, 0)); // Pixel 2: Red
```

```
  strip.setPixelColor(2, strip.Color(0, 0, 255, 0)); // Pixel 3: Blue
 strip.setPixelColor(2, strip.Color(0, 255, 0, 0)); // Pixel 4: Green
  strip.show();

  for (int t = 0; t < 5; t++) {
    tone(BUZZER_PIN, tones[t]);
    delay(150);
    noTone(BUZZER_PIN);
    delay(50);
  }

  delay(10000); // 10 second delay before the loop repeats
}

void func_meas_off(){
  dac.setVoltage(0, false);
  iCounter_off = 0;
  V_VRM_off_v = 0.0;
  I_sense_off_A = 0.0;
  itime_stop_usec = micros() + itime_off_msec * 1000;
  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;
}

void func_meas_on(){
  I_DAC_ADU = I_A * R_sense * DAC_ADU_per_v;
  dac.setVoltage(I_DAC_ADU, false);
  iCounter_on = 0;
  V_VRM_on_v = 0.0;
  I_sense_on_A = 0.00;
  itime_stop_usec = micros() + itime_on_msec * 1000;
  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);
  V_VRM_on_v = V_VRM_on_v / iCounter_on;
  I_sense_on_A = I_sense_on_A / iCounter_on;
}
```