Board 4: A 4-layer Instrument Droid board Report

Name: Pradyumna Gudluru **Date:** 05-08-2023

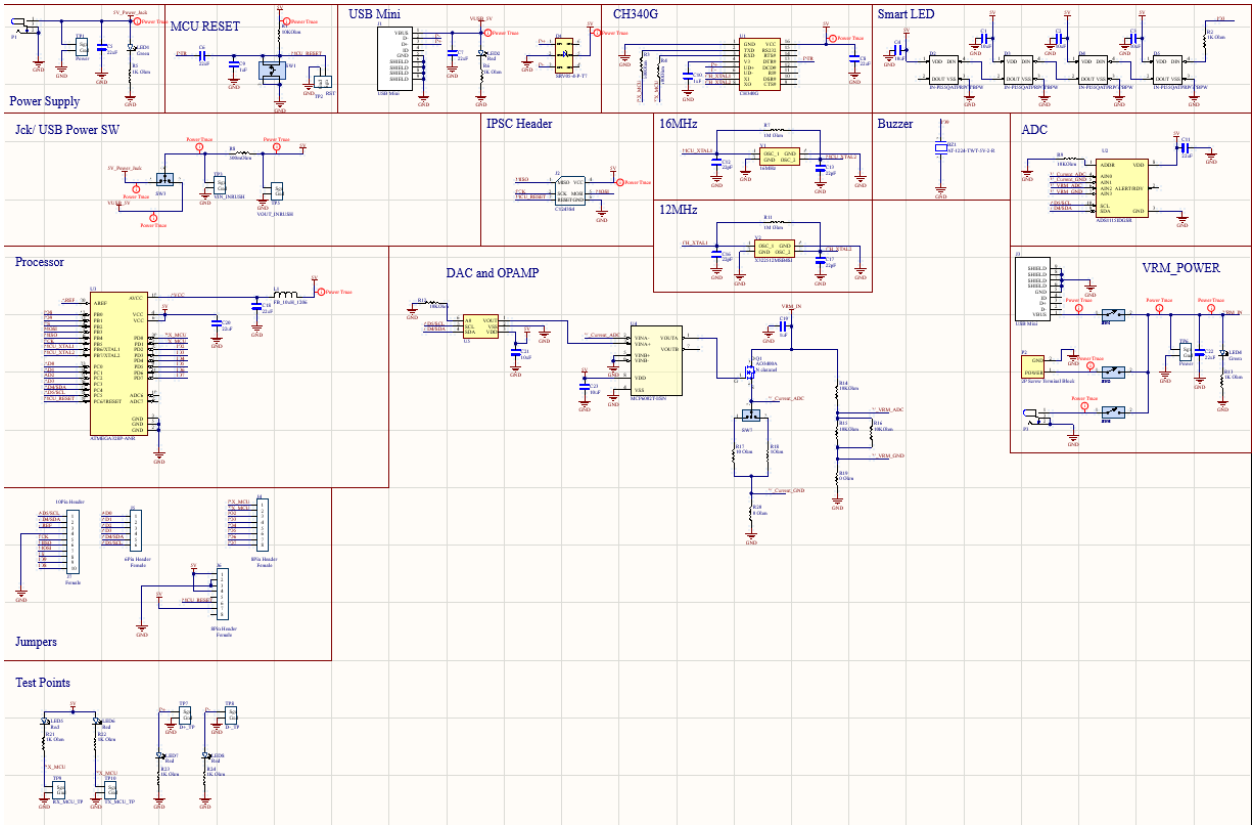
Purpose: The purpose of this board is to design and build a 4-layer board with many different components and finally perform a special function of estimating the resistance of input sources with some code written on Arduino.

POR:

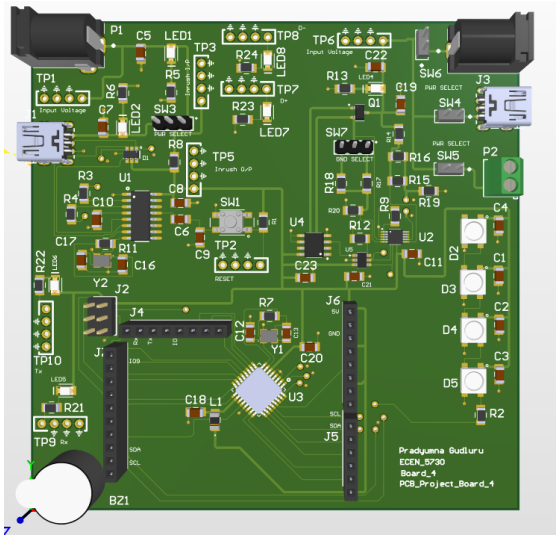
The features that are implemented in the board 4 are,

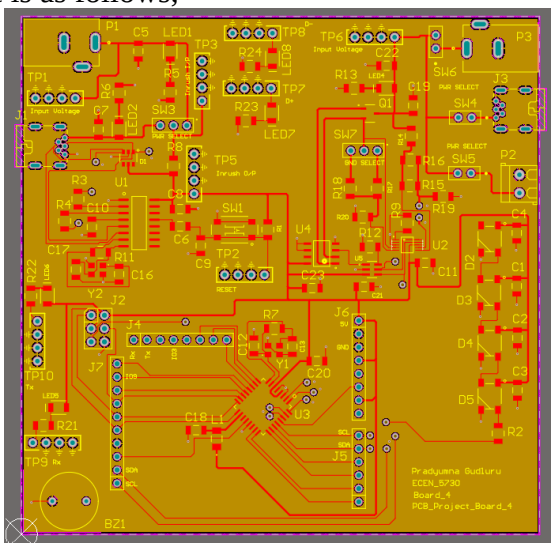
1. It is a 4-layered board and a 328 microcontroller for Arduino to work.
2. Smart LEDs and buzzers are connected as an indication to the start and stop of current pulse.
3. A MCP4725 DAC, MCP 6002 Op Amp, ADC1115 ADC and a MOSFET are used for the instrument droid.
4. The current through the VRM source is enabled with 10% duty cycle which keeps the parts away from duty cycle.
5. The current is ramped up in steps from 0A to 3A.
6. The current ramp up is stopped when the voltage on the VRM drops less than 75% of the unloaded voltage.
7. The range of VRM voltage used is 3.3V - 10V.
8. The VRM voltage unloaded and loaded is measured across the resistors as a differential voltage.
9. The Thevenin's resistance, the current load, the voltage loaded are plotted using an excel sheet.
10. The input of VRM can be either given by USB, power jack or screw terminal.
11. The VRM under test is connected to local ground on the board.

Based on the above requirements, the schematic design for my instrument droid board is as follows,

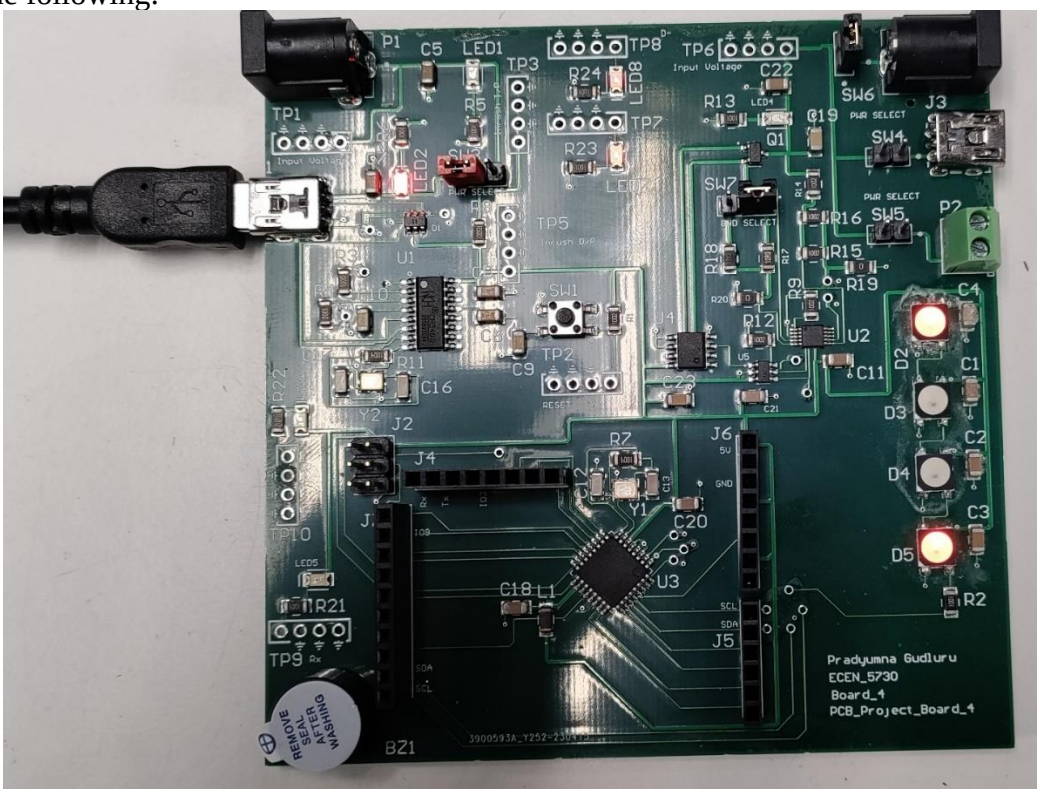


The layout and the 3D model for the above schematic is as follows,





After getting the general PCB from JLCPCB the board after soldering all the components looks like the following:



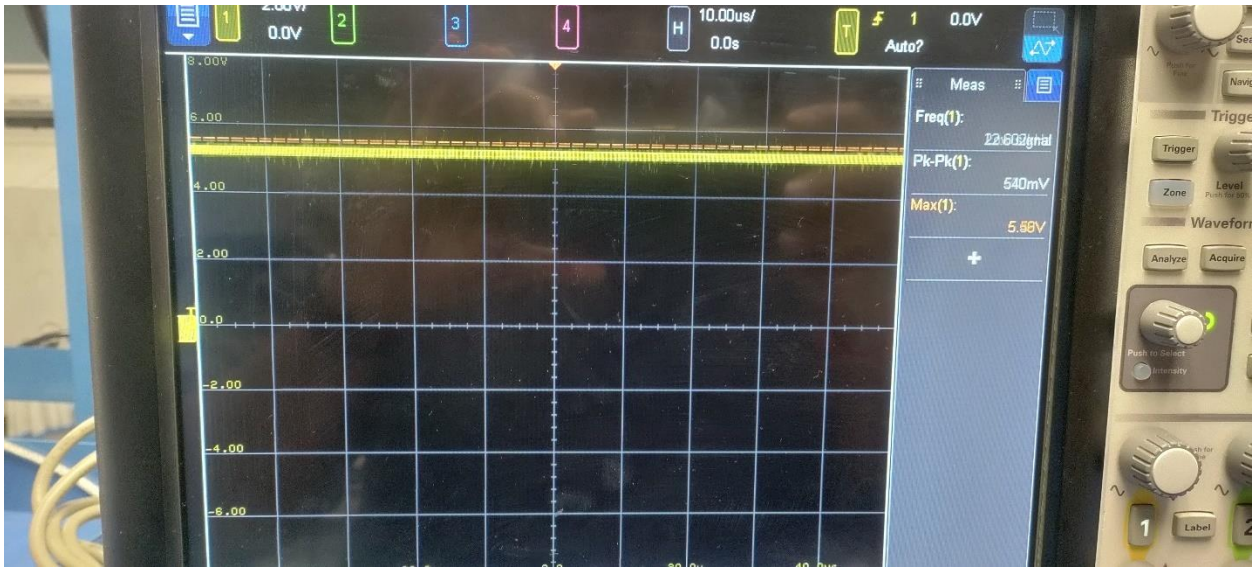
Working:

The assembled board is said to be working if, the LEDs on the input power is ON, The Tx and Rx LED's glow as per the functionality, The Smart LEDs switch ON as per the code and the buzzer switches ON as per the code.

As per the above statements the board worked. It is verified by a step-by-step process as follows.

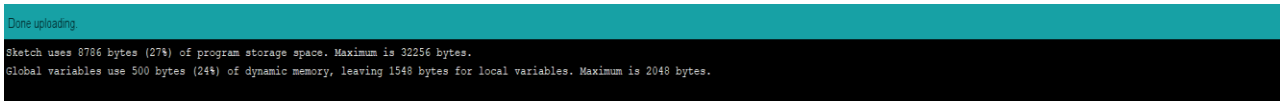
1. **Input Power 5V through the power Jack or USB**

   The 5V input voltage is observed on the oscilloscope from the test point 1 and the LED I slighted up.



2. Boot loading of Arduino

The Arduino is boot loaded and the code is uploaded on to the instrument droid. The following screenshot can be considered.



3. Smart LED and Buzzer Working

The LEDs are lighted up using the example code given by neopixel as per the lab manual. The buzzer is switched on and off in a loop as per the SBB version of Lab 24. In my instrument droid I have used IO3 for controlling the LEDs (Digital Pin 3) and IO9 for the control of buzzer (Digital Pin 9). The pins are probed on the instrument droid to verify the signals. The code is updated in the Appendix. The following is a screenshot of LEDs blinking.

The following are screenshots of the control signals of Pin #3 and Pin #9.

Fig: Probes for IO9 and IO3 digital pins

Fig: Digital Pin #3 for controlling the smart LEDs.

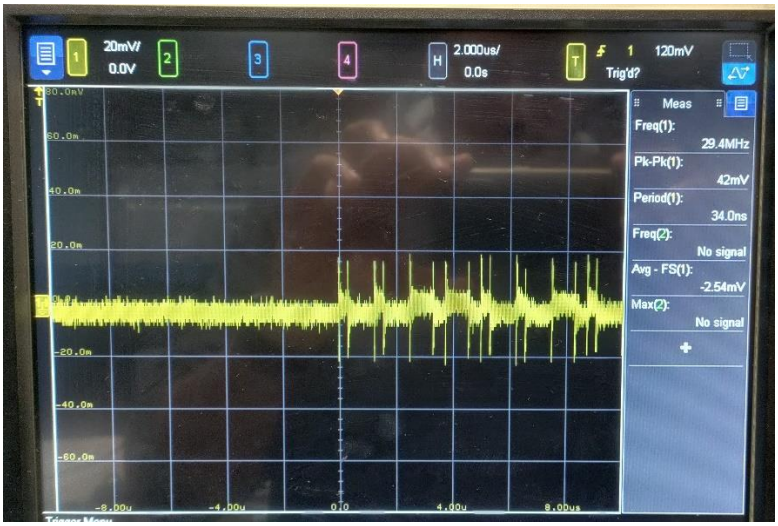
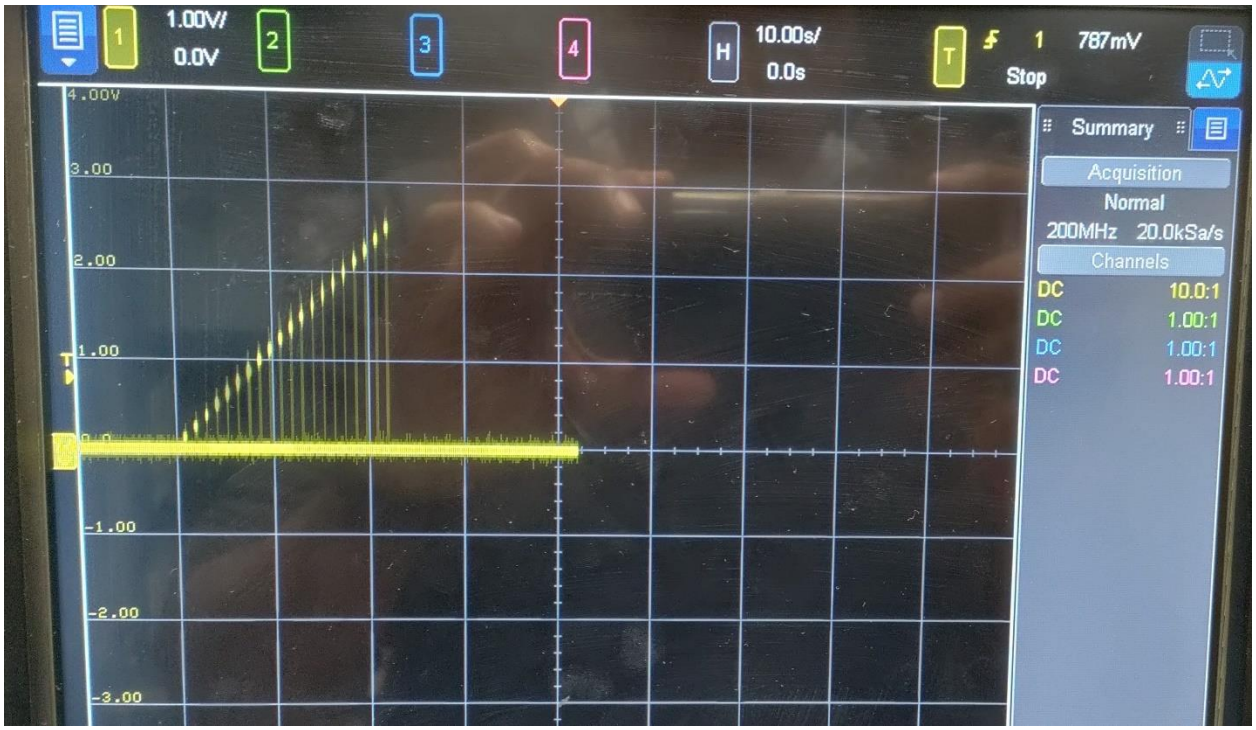


Fig: Digital Pin #9 for controlling Buzzer

4. DAC output

The DAC output is observed by probing on to the pad of the DAC and the following is the screenshot observed on the Oscilloscope.



5. Serial printing of VRM voltages and Thevenin's resistance calculation

Based on different VRM voltages, the Thevenin's resistance is calculated. The ADC values can be observed on the serial monitor of the COM port of instrument droid. The DAC MCP4725 is used to generate a voltage which will match the voltage across the sense resistor. The voltage across the sense resistor and voltage across the potential divider circuit is measured with a 16-bit ADC, ADS1115. A MOSFET is used for switching as the current passing through will be same as in sense resistor.

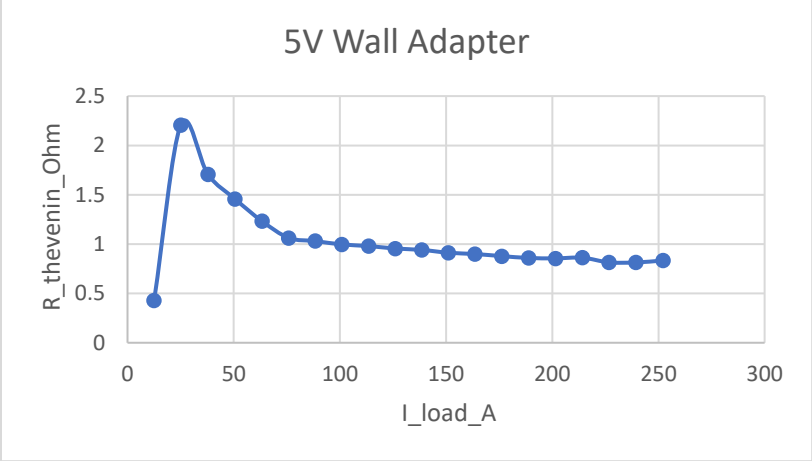
With the Arduino code provided by the professor in the lab manual, for different values of VRM are observed as follows:

- **VRM = 5V input from wall adapter**

The VRM voltage is given to the circuit by a 5V DC supply adapter. The estimated R_Thevenin is around the range of 0.42Ohm to 2.2Ohm. The following is the table for the values.

Sample No.	I_load_A	V_VRM_thevenin_v	V_VRM_loaded_v	R_thevenin
1	12.522	5.3561	5.3508	0.4297
2	25.114	5.3562	5.3008	2.2077
3	37.868	5.3561	5.2915	1.7063
4	50.585	5.3562	5.2825	1.4577
5	63.382	5.3561	5.2781	1.2317
6	75.954	5.3544	5.2737	1.0624
7	88.393	5.3545	5.2634	1.0306
8	100.959	5.3553	5.2547	0.9963
9	113.503	5.3568	5.2456	0.9802
10	126.063	5.3569	5.2365	0.9548
11	138.614	5.3554	5.2249	0.9412
12	151.082	5.3554	5.2176	0.9121
13	163.626	5.3554	5.2082	0.8995
14	176.236	5.3554	5.2007	0.878
15	188.855	5.3567	5.1944	0.8594
16	201.543	5.357	5.1846	0.8554
17	214.209	5.356	5.1713	0.862
18	226.705	5.3569	5.1722	0.8147
19	239.414	5.3556	5.1608	0.8137
20	252.229	5.3567	5.1463	0.834

The sample plot is given by,

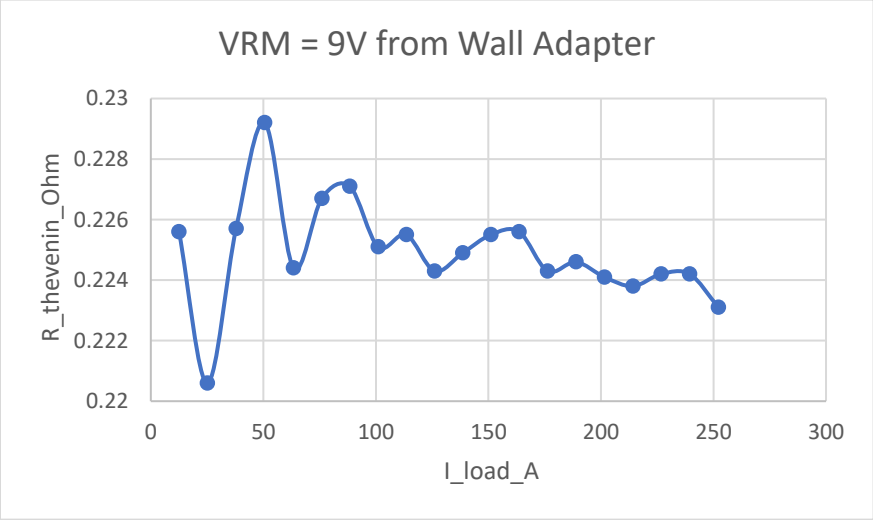


- **VRM = 9V input from wall adapter**

The VRM voltage is given to the circuit by a 9V DC supply adapter. The estimated R_Thevenin is around the range of 0.22Ohm. The following is the table for the values.

Sample No.	I_load_A	V_VRM_thevenin_v	V_VRM_loaded_v	R_thevenin
1	12.519	9.2732	9.2703	0.2256
2	25.12	9.2733	9.2678	0.2206
3	37.87	9.2734	9.2648	0.2257
4	50.59	9.2735	9.2619	0.2292
5	63.39	9.2735	9.2593	0.2244
6	75.965	9.2735	9.2563	0.2267
7	88.4	9.2736	9.2535	0.2271
8	100.969	9.2737	9.251	0.2251
9	113.507	9.2739	9.2483	0.2255
10	126.067	9.2738	9.2455	0.2243
11	138.633	9.2739	9.2428	0.2249
12	151.091	9.2739	9.2398	0.2255
13	163.609	9.2739	9.237	0.2256
14	176.249	9.274	9.2344	0.2243
15	188.857	9.274	9.2316	0.2246
16	201.544	9.2741	9.2289	0.2241
17	214.203	9.2741	9.2262	0.2238
18	226.693	9.2742	9.2234	0.2242
19	239.413	9.2741	9.2204	0.2242
20	252.235	9.2742	9.2179	0.2231

The sample plot is given by,

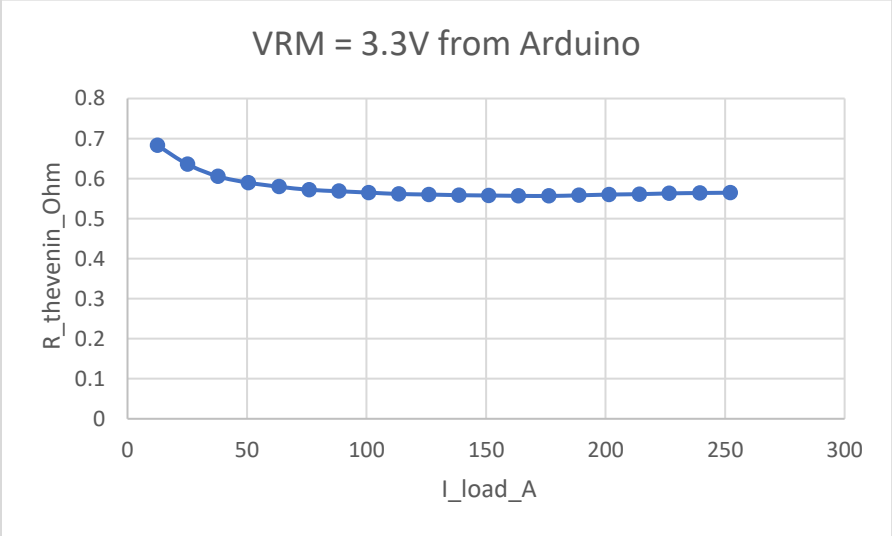


- **VRM = 3.3V input from Arduino**

The VRM voltage is given to the circuit by 3.3V from Arduino. The estimated R_Thevenin is around the range of 0.55Ohm to 0.68Ohm. The following is the table for the values.

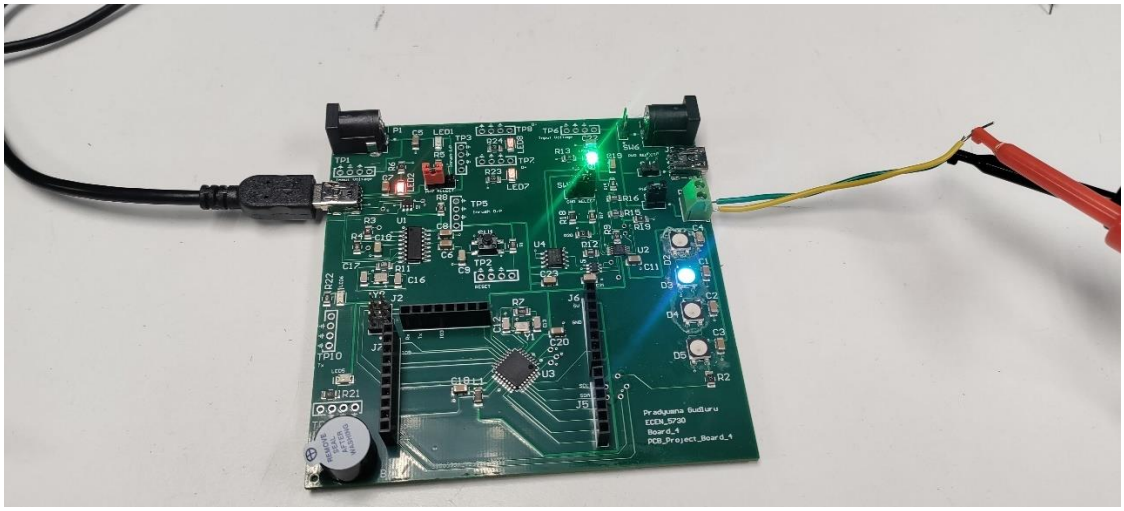
Sample No.	I_load_A	V_VRM_thevenin_v	V_VRM_loaded_v	R_thevenin
1	12.519	3.3078	3.2993	0.6829
2	25.109	3.3079	3.292	0.6357
3	37.861	3.3081	3.2851	0.6056
4	50.577	3.3082	3.2784	0.5898
5	63.374	3.3084	3.2716	0.5797
6	75.942	3.3085	3.2651	0.5721
7	88.379	3.3088	3.2585	0.5684
8	100.938	3.3089	3.2519	0.565
9	113.491	3.3091	3.2453	0.5615
10	126.031	3.3093	3.2387	0.5602
11	138.595	3.3094	3.232	0.5586
12	151.043	3.3096	3.2254	0.5578
13	163.584	3.3098	3.2187	0.5569
14	176.207	3.31	3.212	0.5566
15	188.812	3.3103	3.2049	0.5581
16	201.486	3.3104	3.1976	0.5601
17	214.15	3.3106	3.1905	0.5611
18	226.659	3.3108	3.1832	0.563
19	239.38	3.311	3.176	0.564
20	252.201	3.3112	3.1688	0.5647

The sample plot is given by,



- **VRM = 10V input from function generator**

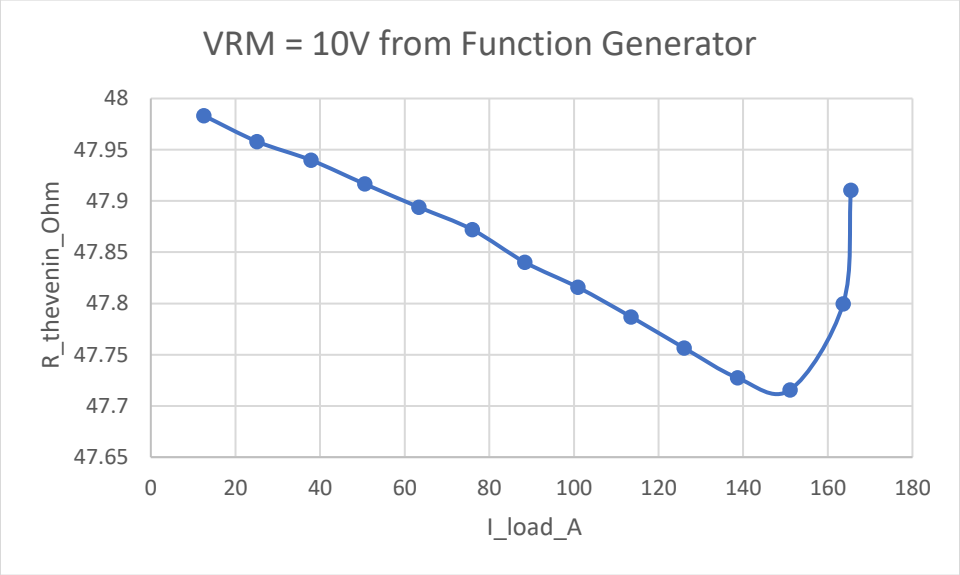
The VRM voltage is given to the circuit by 3.3V from Arduino. The estimated R_Thevenin is around the range of 0.55Ohm to 0.68Ohm. the following is the picture of the board while connected to function generator as VRM.



The following is the table with values.

Sample No.	I_load_A	V_VRM_thevenin_v	V_VRM_loaded_v	R_thevenin
1	12.523	9.6769	9.076	47.9833
2	25.117	9.677	8.4724	47.958
3	37.874	9.677	7.8613	47.9398
4	50.594	9.6769	7.2527	47.9165
5	63.384	9.6769	6.6412	47.8939
6	75.956	9.6769	6.0408	47.8721
7	88.404	9.677	5.4477	47.8402
8	100.971	9.677	4.849	47.8157
9	113.51	9.677	4.2526	47.7869
10	126.069	9.677	3.6564	47.7565
11	138.646	9.6769	3.0597	47.7274
12	151.091	9.677	2.4676	47.7157
13	163.628	9.6769	1.8556	47.7996
14	165.463	9.6769	1.7496	47.9104
15	165.447	9.677	1.7322	48.0198
16	165.445	9.677	1.7166	48.1151
17	165.444	9.6769	1.7051	48.1843
18	165.441	9.677	1.6968	48.2358
19	165.44	9.677	1.6904	48.2749
20	165.443	9.6769	1.6853	48.3043

The sample plot is given by,



The following is the screenshot from the serial monitor.

```
1, 12.523, 9.6769, 9.0760, 47.9833
2, 25.117, 9.6770, 8.4724, 47.9580
3, 37.874, 9.6770, 7.8613, 47.9398
4, 50.594, 9.6769, 7.2527, 47.9165
5, 63.384, 9.6769, 6.6412, 47.8939
6, 75.956, 9.6769, 6.0408, 47.8721
7, 88.404, 9.6770, 5.4477, 47.8402
8, 100.971, 9.6770, 4.8490, 47.8157
9, 113.510, 9.6770, 4.2526, 47.7869
10, 126.069, 9.6770, 3.6564, 47.7565
11, 138.646, 9.6769, 3.0597, 47.7274
12, 151.091, 9.6770, 2.4676, 47.7157
13, 163.628, 9.6769, 1.8556, 47.7996
14, 165.463, 9.6769, 1.7496, 47.9104
15, 165.447, 9.6770, 1.7322, 48.0198
16, 165.445, 9.6770, 1.7166, 48.1151
17, 165.444, 9.6769, 1.7051, 48.1843
18, 165.441, 9.6770, 1.6968, 48.2358
19, 165.440, 9.6770, 1.6904, 48.2749
20, 165.443, 9.6769, 1.6853, 48.3043
done
```

Conclusion:

According to the system, with a sense resistance of 1Ohm on the VRM side, the resistor started to have fumes. I have shifted the switch to connect with the ground through a 10Ohm sense resistance, which made the circuit work with different VRMs. Hence, I would change the resistance based on this experiment next time. I would also increase the resistance, so it can take the values of higher voltages for larger Thevenin's resistance.

I would also like to add a couple of test points on the ADC input and output, DAC input and output so that it would be easy to debug if there is a chance of error. The test point or LED which can show the switching of MOSFET would also be helpful. For basic program (LED blinking on Digital Pin #13) an LED on pin #13 would help in easy verification of Arduino working.

The MOSFET switching worked with current values and Smart LEDs and Buzzer control worked. The noise reduction method on a 4 layer board, by adding a ground via near to a signal transfer from top layer to bottom layer, helped in reduction of noise on signal paths. Following the best design practices and measurement practices helped to reduce the noise and increase signal efficiency.

There were no hard errors noted on the board except the 1 Ohm resistance value. In the design of Smart LEDs and buzzer, I would have added a switch to isolate the working of LEDs so that it doesn't glow continuously if the code running is not related to LEDs.

Some of the best design practices like isolating switches, connecting the ground plane on the layout, identifying the blocks of orientation on schematic helped is understanding and profiling signals on the PCB. Labelling every component and the test points made soldering and probing easier. The signals are probed using the spring tip edges which reduce the noise while signal transmission of profiling.

References:

1. Lab manual provided by Prof. Eric Bogatin
2. https://sites.google.com/colorado.edu/practicalpcbdesignmanufacture/erics-altium-workshop

Appendix:

Code Used for VRM calculation from Manual.

```
// vrm characterizer board
#include <Wire.h>
#include <Adafruit_MCP4725.h>
#include <Adafruit_ADS1X15.h>
Adafruit_ADS1115 ads;
Adafruit_MCP4725 dac;

float R_sense = 10.0; //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 scale
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() {
  Serial.begin(115200);
  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(); // note- you can put the address of the ADS111 here if needed
  ads.setDataRate(RATE_ADS1115_860SPS);// sets the ADS1115 for higher speed
}

void loop() {
  for (i = 1; i <= npts; i++)
  {
    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 < 0.75 * V_VRM_thevenin_v){
      i = npts; //stops the ramping
    }

    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");
    delay(30000);
}

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_2_3() * ADC_V_per_ADU / v_divider + V_VRM_off_v;
    I_sense_off_A = ads.readADC_Differential_0_1() * 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_2_3() * ADC_V_per_ADU / v_divider + V_VRM_on_v;
    I_sense_on_A = ads.readADC_Differential_0_1() * 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);
}
```