Practical PCB Design and Manufacture

Lab-21,22 Report: SBB Version of Board 4 and its Applications

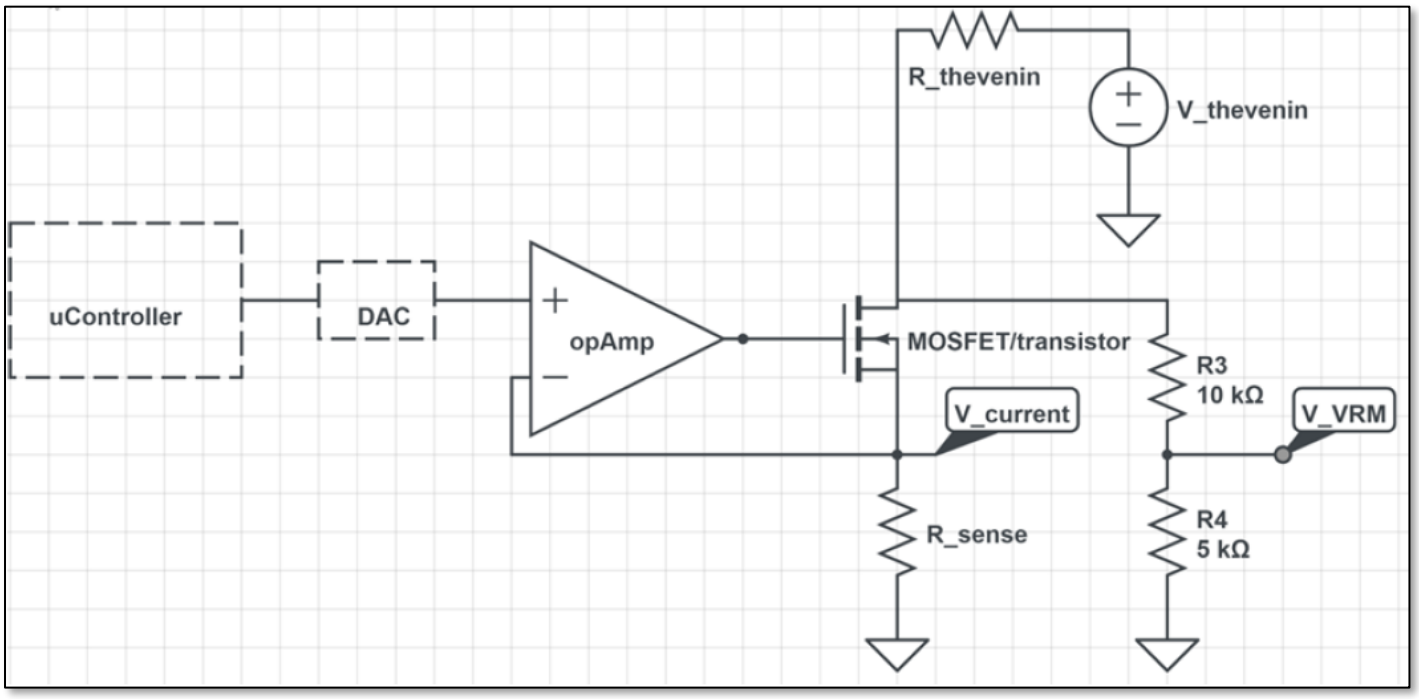
Objective / Purpose of the Lab:

- The aim of this lab is to build an SBB version of the instrument droid. The instrument droid is an intelligent measurement system designed to characterize any voltage source, or voltage regulator module (VRM) by measuring its Thevenin voltage and the Thevenin resistance as a function of output current. With a current load, the voltage drop is the voltage across the internal Thevenin resistor. Knowing the voltage drop and the current through it, we calculate the Thevenin resistance.

Component Listing:

- Solderless Bread Board
- Arduino UNO
- MCP4725 DAC
- T2721 Op-Amp
- IRL520 MOSFET
- ADS1115 ADC
- Connecting Wires
- Oscilloscope (With 10x Probes)
- Digital Multimeter
- Capacitors
- Resistors

Board Connections Sketch:



Code Implemented:

```
// vrm characterizer board
#include <Wire.h>
#include <Adafruit_MCP4725.h>

Adafruit_MCP4725 dac;

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

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
}

void loop()
{
  V_DAC_ADU = 0 * DAC_ADU_per_v;
  dac.setVoltage(V_DAC_ADU, false); //sets the output current to 0 initially

  V_DAC_ADU = 3.0 * DAC_ADU_per_v;
  dac.setVoltage(V_DAC_ADU, false); //sets the output current to 0 initially
}
```

The code above is implemented for testing the DAC output. We get a pulse wave on the oscilloscope. We measure the voltage across the voltage divider of the VRM with channel A0 and A1 and across the sense resistor with channels A2 and A3.

The code given below is designed for a Voltage Regulator Module (VRM) characterization system using an Arduino platform. It employs Adafruit's MCP4725 DAC and ADS1115 ADC to generate and measure voltages and currents. The system operates by sequentially increasing load currents, measuring voltage drops across a sense resistor and the VRM to calculate the VRM's Thevenin resistance and load characteristics. This process helps in assessing the VRM's performance under different load conditions.

```ino
// vrm characterizer board
#include <Wire.h>
#include <Adafruit_MCP4725.h>
#include <Adafruit_ADS1X15.h>

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()
{
  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(0x48); // 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 < 0.25 * 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_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);
}
```

Code Output and Graph Plotted:

When connected to a 9V as V_thevenin power supply and directly connected the input to the drain of the IRL520n MOSFET, we get the following LOGs on the terminal window.

```
, 8.068, 8.5713, 8.5699, 0.1801
, 16.172, 8.5715, 8.5683, 0.1946
, 24.414, 8.5715, 8.5666, 0.2008
, 32.627, 8.5715, 8.5633, 0.2512
, 40.830, 8.5715, 8.5617, 0.2392
, 49.012, 8.5715, 8.5591, 0.2529
, 57.314, 8.5715, 8.5580, 0.2353
, 65.518, 8.5716, 8.5576, 0.2130
, 73.426, 8.5716, 8.5552, 0.2229
, 81.546, 8.5717, 8.5528, 0.2310
, 89.758, 8.5716, 8.5531, 0.2061
, 98.027, 8.5716, 8.5529, 0.1904
, 106.147, 8.5716, 8.5513, 0.1916
, 108.986, 8.5717, 8.5511, 0.1889
, 109.003, 8.5717, 8.5508, 0.1910
, 109.018, 8.5717, 8.5511, 0.1884
, 109.031, 8.5717, 8.5502, 0.1967
, 109.048, 8.5716, 8.5514, 0.1858
, 109.067, 8.5718, 8.5518, 0.1826
, 109.083, 8.5717, 8.5521, 0.1804
done
, 8.061, 8.5718, 8.5703, 0.1849
```

Fig – 1: 9V Power Supply ADC O/P

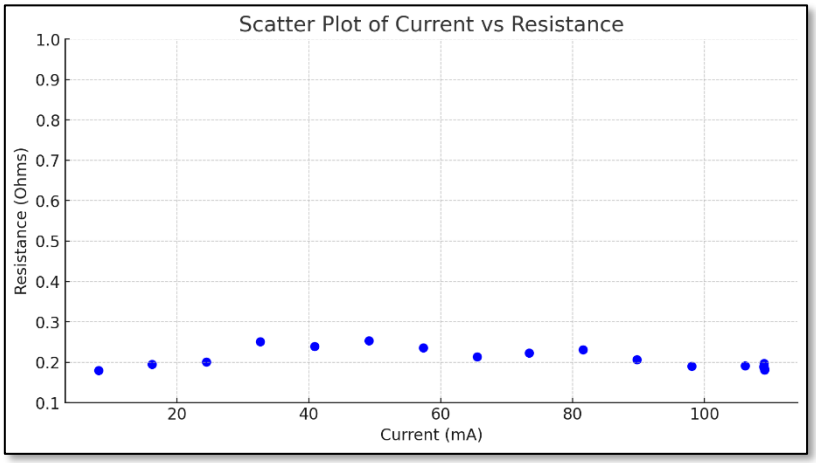


Fig – 2: 9V ADC O/P Scatter Plot

When a 10Ω Resistor is connected as R_Thevenin between the 9V power supply and the drain of the MOSFET, the resistance gets multipled almost by a factor of 10 as shown in Fig-3.

```
, 8.054, 8.5640, 8.4851, 9.8051
, 16.173, 8.5641, 8.4049, 9.8472
, 24.414, 8.5641, 8.3241, 9.8323
, 32.442, 8.5642, 8.2454, 9.8251
, 40.586, 8.5641, 8.1654, 9.8235
, 48.811, 8.5641, 8.0848, 9.8214
, 57.024, 8.5641, 8.0042, 9.8191
, 65.248, 8.5642, 7.9237, 9.8162
, 73.301, 8.5642, 7.8448, 9.8155
, 81.555, 8.5642, 7.7642, 9.8090
, 89.820, 8.5642, 7.6826, 9.8149
, 98.015, 8.5642, 7.6021, 9.8157
, 106.148, 8.5641, 7.5227, 9.8118
, 109.173, 8.5642, 7.4914, 9.8263
, 109.179, 8.5641, 7.4899, 9.8393
, 109.189, 8.5641, 7.4911, 9.8275
, 109.199, 8.5641, 7.4908, 9.8294
, 109.210, 8.5642, 7.4906, 9.8303
, 109.223, 8.5641, 7.4898, 9.8360
, 109.224, 8.5641, 7.4908, 9.8268
done
, 8.067, 8.5642, 8.4849, 9.8285
```

Fig – 3: 9V Power Supply with 10Ω ADC O/P

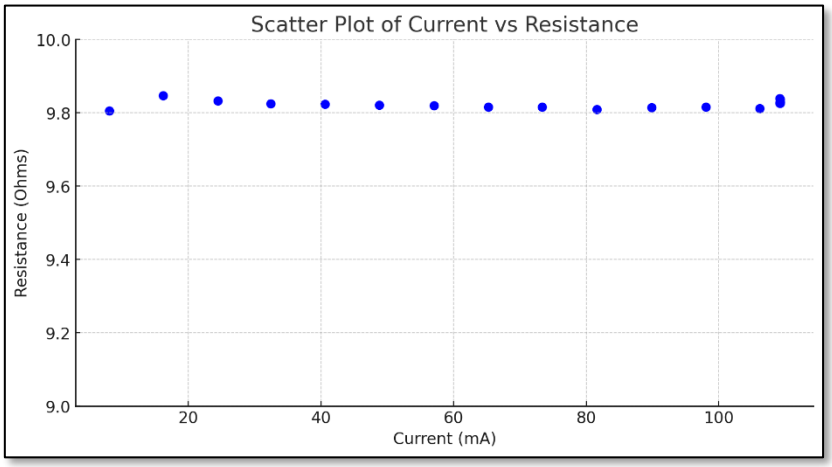


Fig – 4: 9V with 10Ω ADC O/P Scatter Plot

When we give a 9V peak-to-peak output from the function generator to the input of the MOSFET at the drain, we see a variety of readings as shown below.

```
, 5.257, 2.2748, 2.1882, 16.4694
, 5.924, 2.2972, 1.9333, 61.4338
, 13.647, 2.2697, 1.7200, 40.2803
, 12.618, 2.3019, 1.6266, 53.5200
, 19.949, 2.2739, 1.2757, 50.0339
, 20.801, 2.2827, 1.3363, 45.4974
, 24.744, 2.2989, 1.0079, 52.1724
, 28.844, 2.2748, 0.9885, 44.5962
, 27.568, 2.3001, 0.8527, 52.5021
, 35.573, 2.2752, 0.5927, 47.2975
, 34.984, 2.2824, 0.6208, 47.4961
, 36.637, 2.2963, 0.4372, 50.7432
, 43.647, 2.2769, 0.3022, 45.2428
, 36.846, 2.2940, 0.3685, 52.2581
, 42.121, 2.2834, 0.2596, 48.0480
, 42.220, 2.2760, 0.3387, 45.8852
, 37.446, 2.3039, 0.3617, 51.8675
, 43.581, 2.2727, 0.2771, 45.7912
, 40.221, 2.2889, 0.3700, 47.7079
, 39.069, 2.2969, 0.3120, 50.8054
done
```

Fig – 5: Function Generator O/P

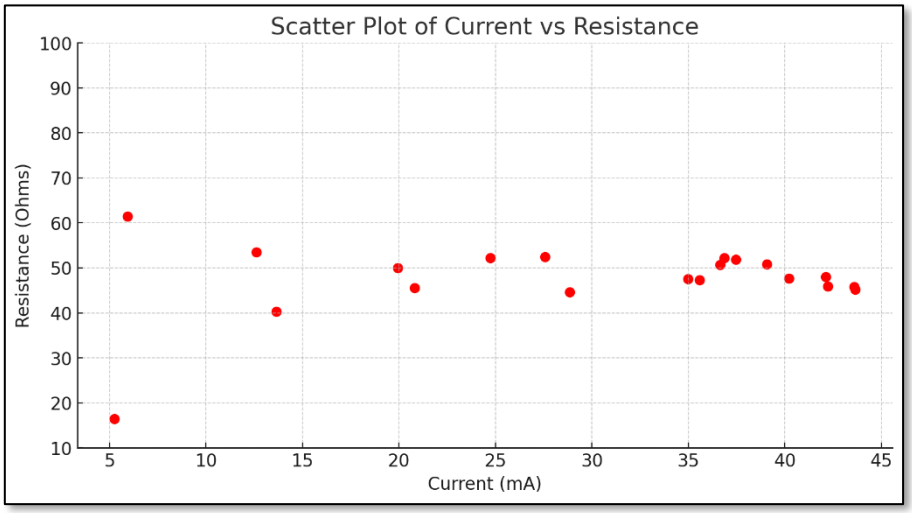


Fig – 6: Function Generator O/P Scatter Plot

Conclusion / Inference:

- The experiment effectively demonstrated how to calculate the Rth value, a key parameter for characterizing power sources.
- We successfully calibrated and measured the Rth for the laboratory function generator and power supplies 9V and 5V.
- Observations were made on the individual outputs from the DAC, SCL, and SDA.