Lab 22 Report
SBB Version of Instrument Droid, Board-4

- Vaishnavi Patekar
vaishnavi.patekar@colorado.edu

Objective

To build an SBB version of an intelligent measurement system called an instrument droid which is specifically 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.

Circuit Diagram

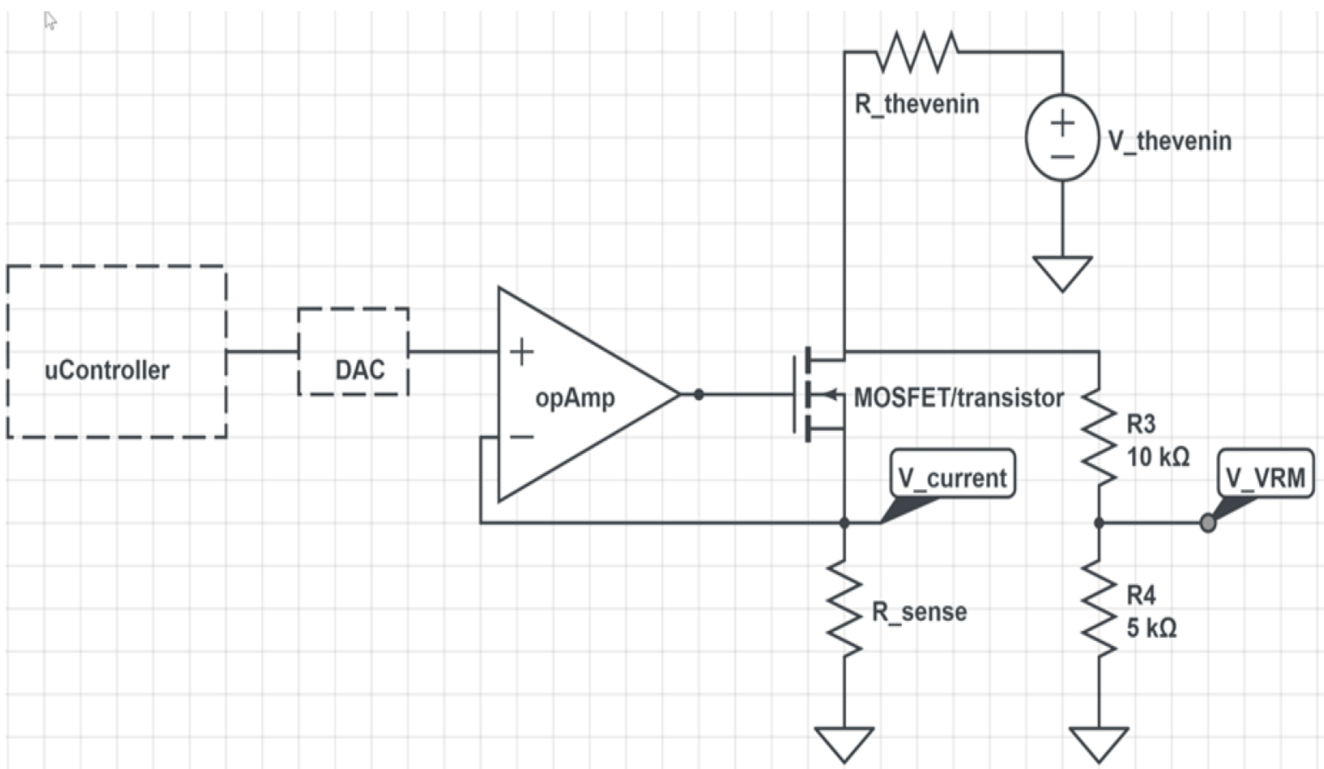

Figure 1: Circuit to measure the VRM characteristics

Components Required

Sr. No.	Component	Function
1.	DAC, MCP4725	To generate a voltage that will match the voltage across the sense resistor
2.	ADC, ADS1115 (16-bit ADC)	To measure the voltage across the voltage divider of the VRM with channels A0 and A1 and across the sense resistor with channels A2 and A3
3.	Op-Amp, MCP602	To drive the transistor across the feedback
4.	MOSFET, IRL520	To measure load Voltage and Un-loaded Voltage When MOSFET ON: Vth MOSFET OFF: V_VRM
5.	Resistors: 10Ω (Sense Resistor), 5kΩ, 10Ω	As a load
6.	Arduino Board	To control ADC, and DAC through I2C
7.	Function Generator	To measure Rth of known instrument

Circuit Setup

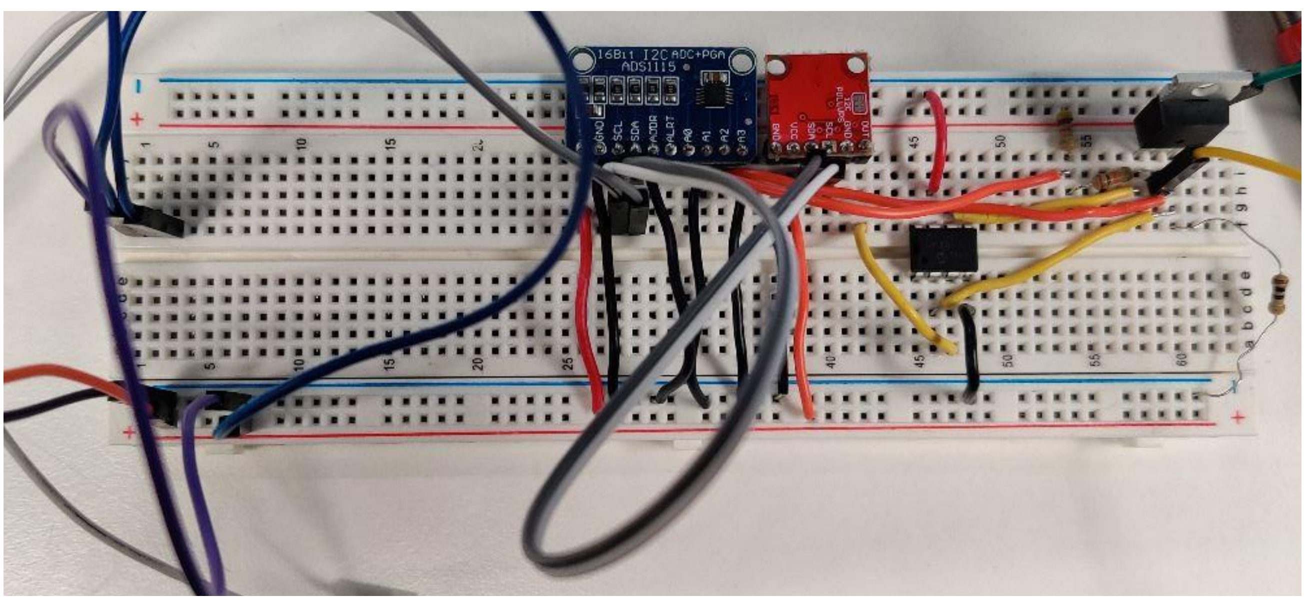

Figure 2: SBB setup for Instrument Droid

Working of a Circuit

- Microcontroller will continuously transmit analog signals through the DAC module connected via the I2C interface, by increasing the step size with each iteration.
- The differential amplifier receives feedback from the FET source as it receives the DAC signal. The transistor can be turned ON or OFF with the aid of this feedback.
- The measured voltage would be VThevenin while the transistor is turned on and VLoaded when it is off. The transistor's Drain terminal is where the VRM's input is provided.

Arduino Code to Run Instrument Droid

```
// 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() {
pinMode(8, OUTPUT);
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() {
digitalWrite (8, HIGH);
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 < 0.50 * 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);
}
```

Analysis

A known resistance was added to a 5V power source to test the circuit's operation and see if the Droid circuit provided the correct value measurement. Other test measurements were done after obtaining the value identical to the increased known resistance with success. Other test measurements were done after successfully obtaining a value equal to the extra known resistance.
It is crucial to keep in mind that in each of the output examples below, the ADC module's channels A0-A1 and A2-A3 are used to take the VLoad and VThevenin, respectively.

Excel graphs of Current vs. Thevenin Resistance supported by the serial console output are as follows:

3.3V from Arduino

```
1, 13.065, 3.1604, 3.1311, 2.2420
2, 25.552, 3.1603, 3.1035, 2.2238
3, 38.092, 3.1603, 3.0753, 2.2313
4, 50.486, 3.1605, 3.0467, 2.2538
5, 62.868, 3.1606, 3.0182, 2.2646
6, 75.201, 3.1607, 2.9899, 2.2718
7, 87.186, 3.1606, 2.9612, 2.2868
8, 99.229, 3.1606, 2.9324, 2.2994
9, 111.125, 3.1606, 2.9014, 2.3328
10, 123.277, 3.1604, 2.8700, 2.3561
11, 135.040, 3.1599, 2.8392, 2.3754
12, 146.720, 3.1600, 2.8078, 2.4008
13, 158.288, 3.1601, 2.7765, 2.4235
14, 169.780, 3.1601, 2.7455, 2.4420
15, 181.403, 3.1603, 2.7136, 2.4621
16, 192.949, 3.1601, 2.6796, 2.4901
17, 204.213, 3.1600, 2.6421, 2.5364
18, 215.307, 3.1600, 2.5994, 2.6040
19, 226.580, 3.1608, 2.5520, 2.6865
20, 237.811, 3.1597, 2.5361, 2.6220
done
```

Figure 3: Serial Console for 3.3V input to the circuit

In figure 3, the data is serially shown on the serial terminal in the following order: Index, current in mA, V_thevenin, V_loaded, and R_thevenin.

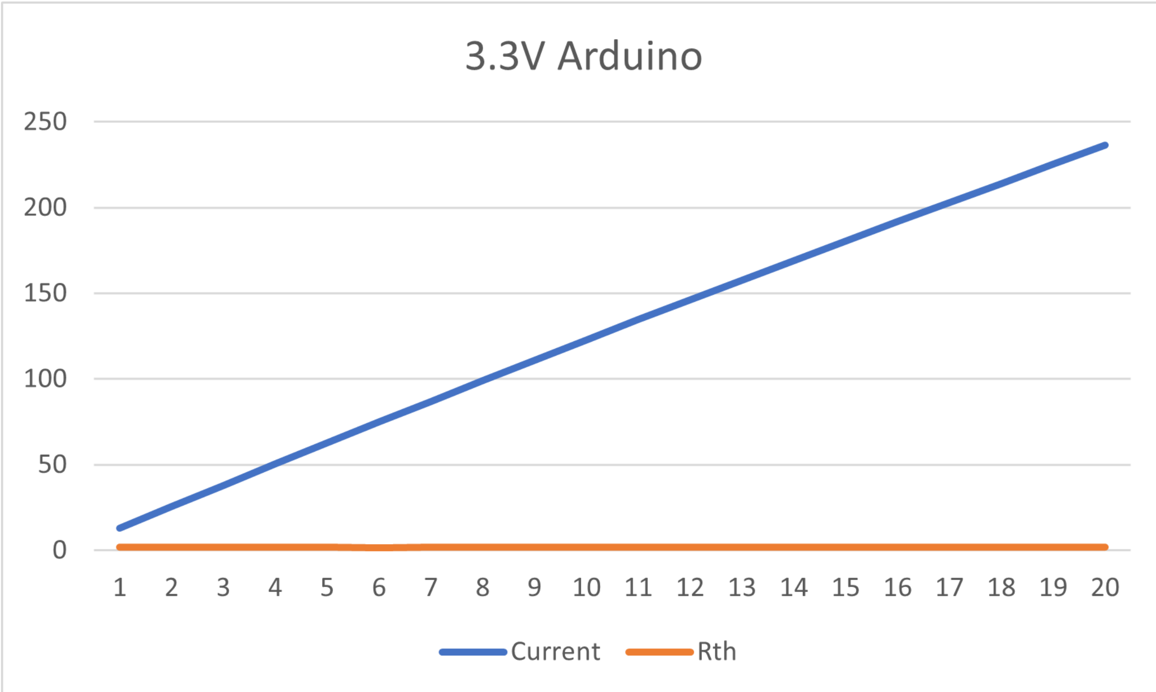

Figure 4: Current vs Rth for 3.3V input

It can be observed that the Thevenin Resistance (RTh) for 3.3V output from Arduino was low, around 2.5186 ohms.

GPIO Output from Arduino (HIGH)

```
1, 13.033, 4.7486, 4.3889, 27.6012
2, 25.528, 4.6591, 4.1267, 20.8537
3, 38.050, 4.6639, 3.7744, 23.3765
4, 50.525, 4.6467, 3.4886, 22.9229
5, 63.028, 4.6848, 3.4132, 20.1755
6, 75.746, 4.6682, 3.6202, 13.8353
7, 88.213, 4.6530, 3.8663, 8.9182
8, 100.201, 4.6891, 3.1306, 15.5531
9, 112.540, 4.6471, 3.0426, 14.2567
10, 125.101, 4.6767, 2.7465, 15.4291
11, 130.002, 4.7458, 1.3558, 26.0764
12, 121.049, 4.7502, 1.1851, 29.4514
13, 111.978, 4.7494, 1.1031, 32.5635
14, 113.753, 4.7469, 1.1282, 31.8120
15, 106.908, 4.7476, 1.0524, 34.5641
16, 107.422, 4.7466, 1.0579, 34.3386
17, 107.926, 4.7469, 1.0619, 34.1444
18, 108.577, 4.7455, 1.0685, 33.8661
19, 110.434, 4.7461, 1.0850, 33.1518
20, 121.145, 4.7467, 1.1892, 29.3655
done
```

Figure 5: Serial Console for GPIO Output from Arduino as input to the circuit

In figure 5, the data is serially shown on the serial terminal in the following order: Index, current in mA, V_thevenin, V_loaded, and R_thevenin.

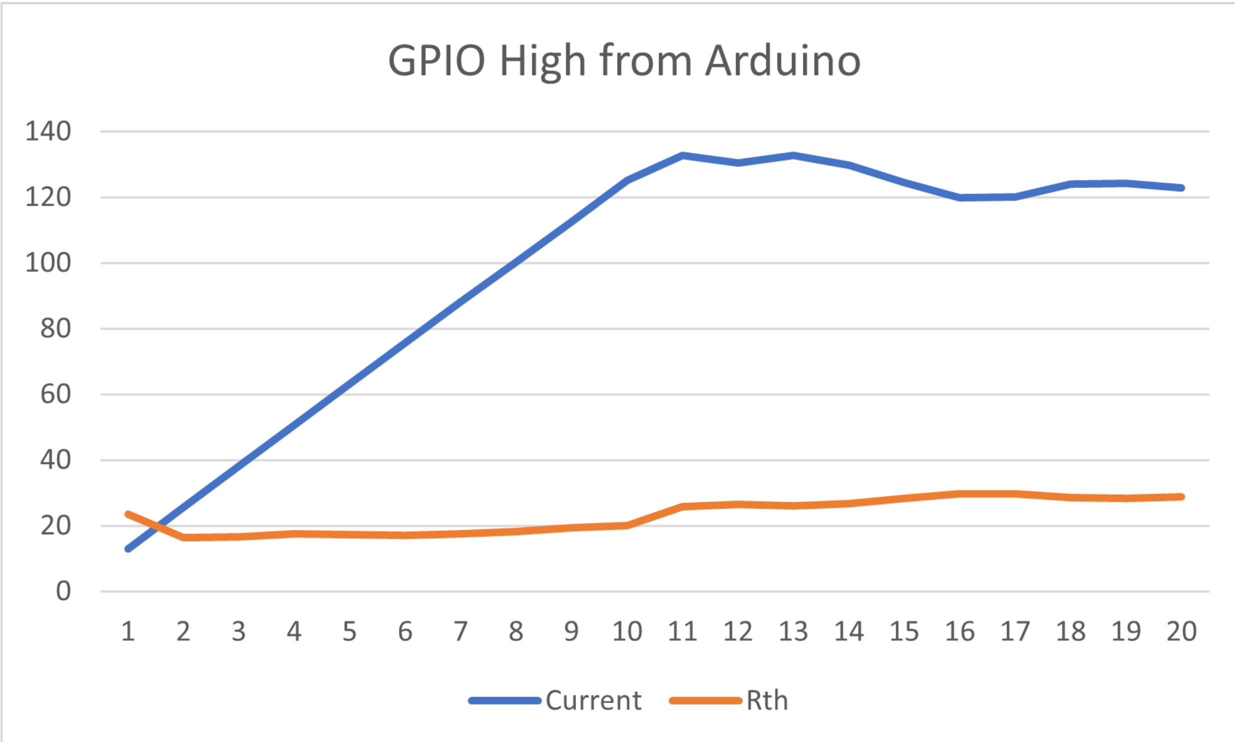

Figure 6: Current vs Rth for GPIO Output from Arduino as input to the circuit

It can be observed from the above traces that Rth is nearly equal to 30Ω.

Function Generator

For earlier verification, Rth for the function generator was measured since we already know Thevinin's resistance for a function generator i.e. 50 Ohms.

Therefore, we should receive the same result when we connect the same to our Instrument Droid Circuit.

The following graph depicts the Current vs. Rth for the connected Function Generator:

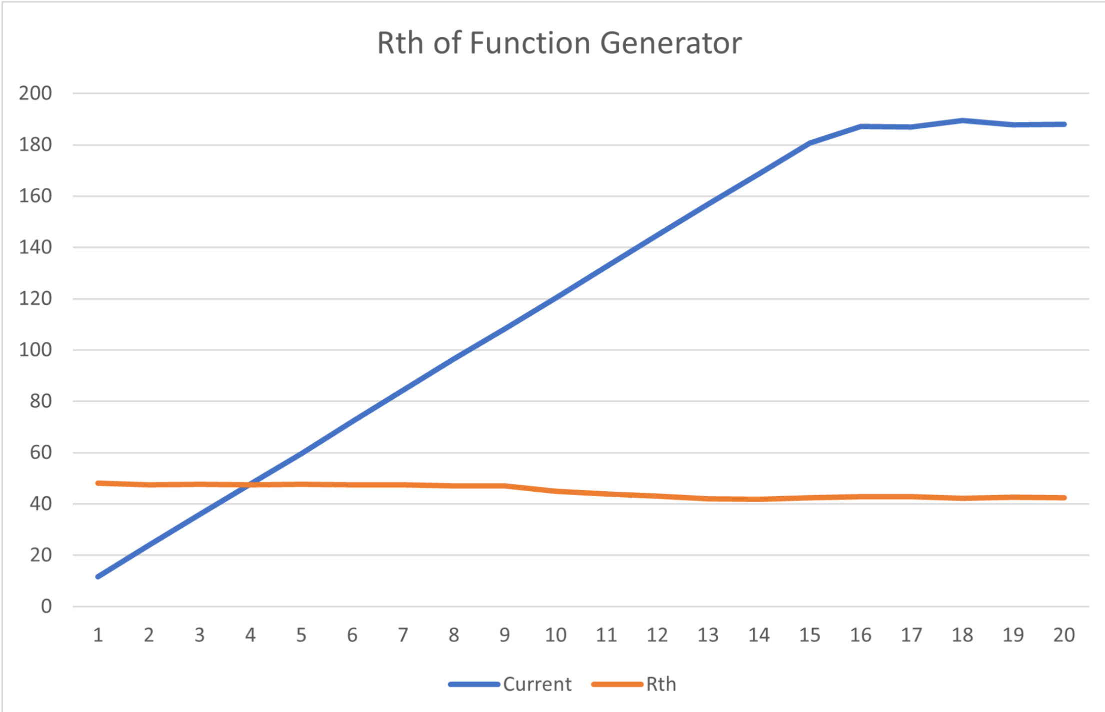

Figure 7: Current vs Rth for Function Generator

It is evident that the instrument droid circuit determined the Thevenin Resistance of the Function Generator to be around 48 Ohms.

Conclusion & Key Learnings

- An instrument droid is a custom instrument specifically 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.
- This lab improved my understanding of how an instrument droid operates, what we can expect from its internal voltages, what we can measure, and some examples of the Thevenin resistance and output current of different sources.
- To confirm the correct functioning of a circuit, it is always better to measure the known parameter like we measured the Thevenin Resistance (Rth) of the Function Generator.
- Before implementing any design on a PCB, it is necessary to bring up and test the SBB version of it.

References

- *CU ECEN-4/5730 Spring 2023 Workbook*
- *Bogatin's Practical Guide to Prototype Breadboard and PCB Design by Eric Bogatin, published by Artech House Copyright: 2021 ISBN: 9781630818487*