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 VVRM when there is no gate voltage applied across the MOSFET.
* Code reads the Load Current IL 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: RTh = (VTh – VL)/IL.

**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.
* 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

A diagram of a circuit

Description automatically generatedA diagram of a circuit board

Description automatically generated**Schematic:**

*Circuit involved in Instrument droid board.*

A screenshot of a computer

Description automatically generatedA screen shot of a video game

Description automatically generatedA screen shot of a video game

Description automatically generatedA yellow circuit board with red and black dots

Description automatically generated**Layout:**

*4 individual layer placement is Highlighted.*

A screenshot of a computer chip

Description automatically generatedA green circuit board with many small chips

Description automatically generated**Board:**

A close up of a circuit board

Description automatically generated

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.

**Scope Output:**

1. **5V power supply:**

A screen shot of a graph

Description automatically generated

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

1. A screen shot of a graph

   Description automatically generatedA screen shot of a computer

   Description automatically generated**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.

1. **USB mini output:**

A screen shot of a computer

Description automatically generated

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

1. **UART output:**

A screen shot of a graph

Description automatically generated

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

1. A screen shot of a graph

   Description automatically generatedA screenshot of a computer

   Description automatically generated**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.

1. **Reset Pin:**

A screen shot of a graph

Description automatically generated

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

1. A screen shot of a graph

   Description automatically generatedA screen shot of a graph

   Description automatically generatedA screen shot of a graph

   Description automatically generated**DAC, opAmp and MOSFET output:**

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.

1. A circuit board with a blue screen and red lights

   Description automatically generatedA screenshot of a computer

   Description automatically generated**Thevenin resistance of 9V battery:**

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

**Plot:  
A graph of a battery

Description automatically generated**

1. A close up of a circuit board

   Description automatically generatedA screenshot of a computer

   Description automatically generated**Thevenin resistance of 9V adapter:**

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:**

**A graph with a line going up

Description automatically generated**

1. A close up of a circuit board

   Description automatically generatedA screenshot of a computer

   Description automatically generated**Thevenin resistance of 5V adapter:**

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:**

**A graph with a line going up

Description automatically generated**

1. A close up of a circuit board

   Description automatically generatedA screenshot of a computer

   Description automatically generated**Buzzer and Smart Led:**

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

1. **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);

}