Low Cost Impedance Meter Project Report

Sean-Michael Woerner – 1001229459

CSE 4342-001

May 4, 2021

Table of Contents

Introduction

The goal of this project is design a system capable of measuring resistance, inductance (and ESR), and capacitance. A design goal of this project is to limit the total cost of the daughterboard and components added to the TM4C123GXL evaluation board to $3 in 10k quantities.

Since connectors have significant associated cost, two shared connections will be used for the device under test, regardless of whether a resistive, an inductive, or a capacitive device is measured, so any commutation must also be included to allow any attached device to be measured.

The project shall provide a complete user interface through the virtual COM port on the evaluation board.

Hardware Description

Microcontroller: An ARM M4F core (TM4C123GH6PMI microcontroller) is required.

Power LED: A power LED must be connected through a current-limiting resister to indicate the daughterboard has power.

Serial interface: If using the EK-TM4C123GXL evaluation board, then the UART0 tx/rx pair is routed to the ICDI that provides a virtual COM port through a USB endpoint.

LCR measurement interface: A circuit is provided that will interface with the microcontroller and allow the user to test an L, C, or R value. The output of this circuit can be connected to the analog comparator and analog-to-digital converter inputs. The circuit will described in detail in class and a schematic are provided. You should know the operation of every component of the circuit and be prepared to answer questions about it on the second exam.

3.3V supply: The circuit is powered completely from the 3.3V regulator output on the evaluation board.

Device under test (DUT) connection; Two connectors, made of wire loop to save cost, are required to allow the DUT to be connected.

Test points: Test points shall be added for the ground reference and comparator output at minimum.

Pushbuttons: 6 pushbuttons in total to initiate: Resistance, Capacitance, Inductance, ESR, Voltage, and Auto commands.

Software Description

A virtual COM power using a 115200 baud, 8N1 protocol with no hardware handshaking shall be provided with support to the following commands.

Debug:

If “reset” is received, the hardware shall reset.

If “voltage” is received, the hardware shall return the voltage across DUT2-DUT1. The voltage is limited to 0 to 3.3V.

LCR commands:

If “resistor” is received, return the resistance of the DUT. You should try to convert a capacitance value from 10ohms to 1Mohm.

If “capacitance” is received, return the capacitance of the DUT. You should try to convert a capacitance value from 1nF to 100uF.

If “inductance” is received, return the inductance of the DUT. You should try to convert an inductance value from 1nH to 100uH.

If “esr” is received, return the ESR of the inductor under test.

If “auto” is received, return the value of the DUT that is most predominant (i.e. an inductor with 1ohm ESR and 10µH inductance will return the inductance and ESR values, a 100kohm resistor will return the resistance, and a 10µF capacitor will return the capacitance.

Pushbuttons:

(1) Measures Resistance

(2) Measures Capacitance

(3) Measures Inductance

(4) Measures ESR

(5) Does Auto

(6) Measures Voltage

Schematic

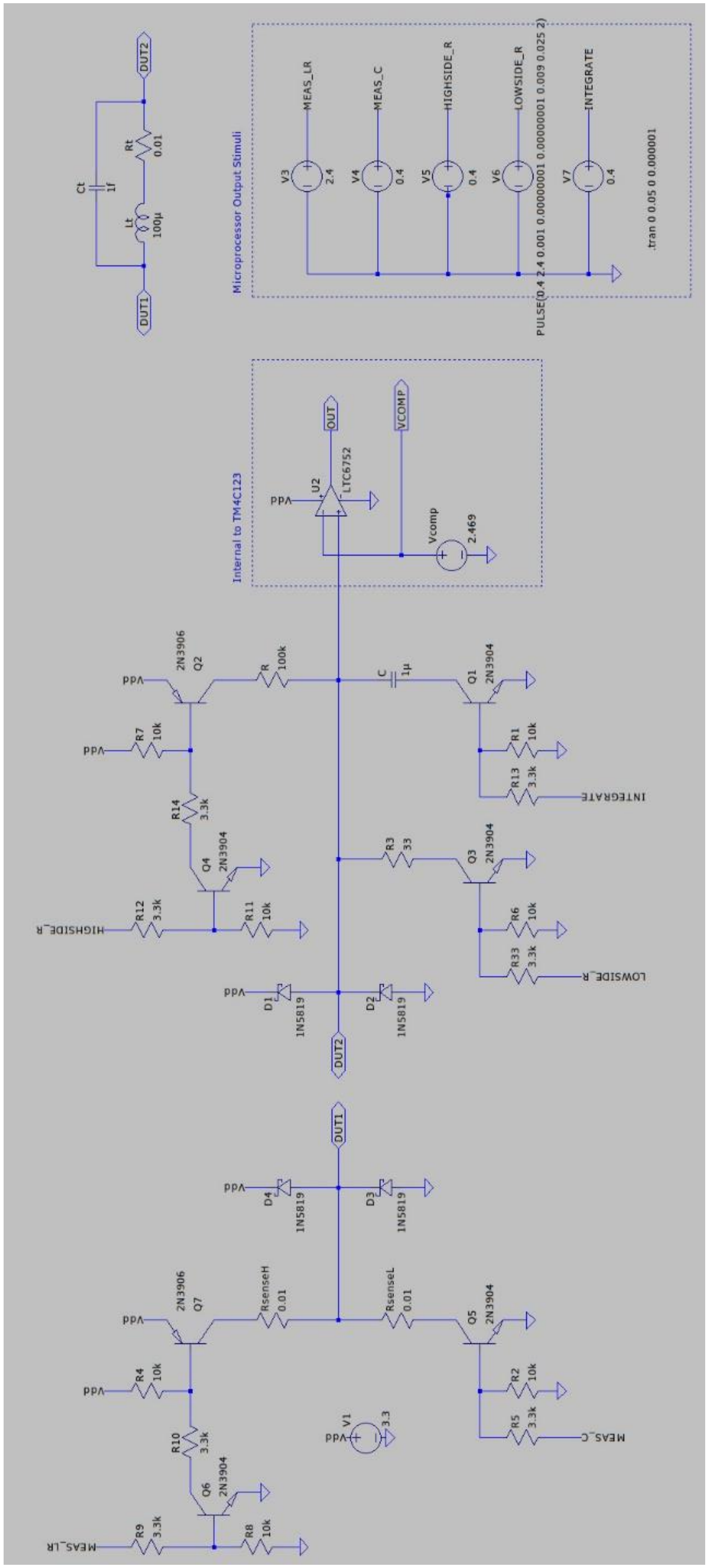


Figure 1 – Project Schematic

System Pictures

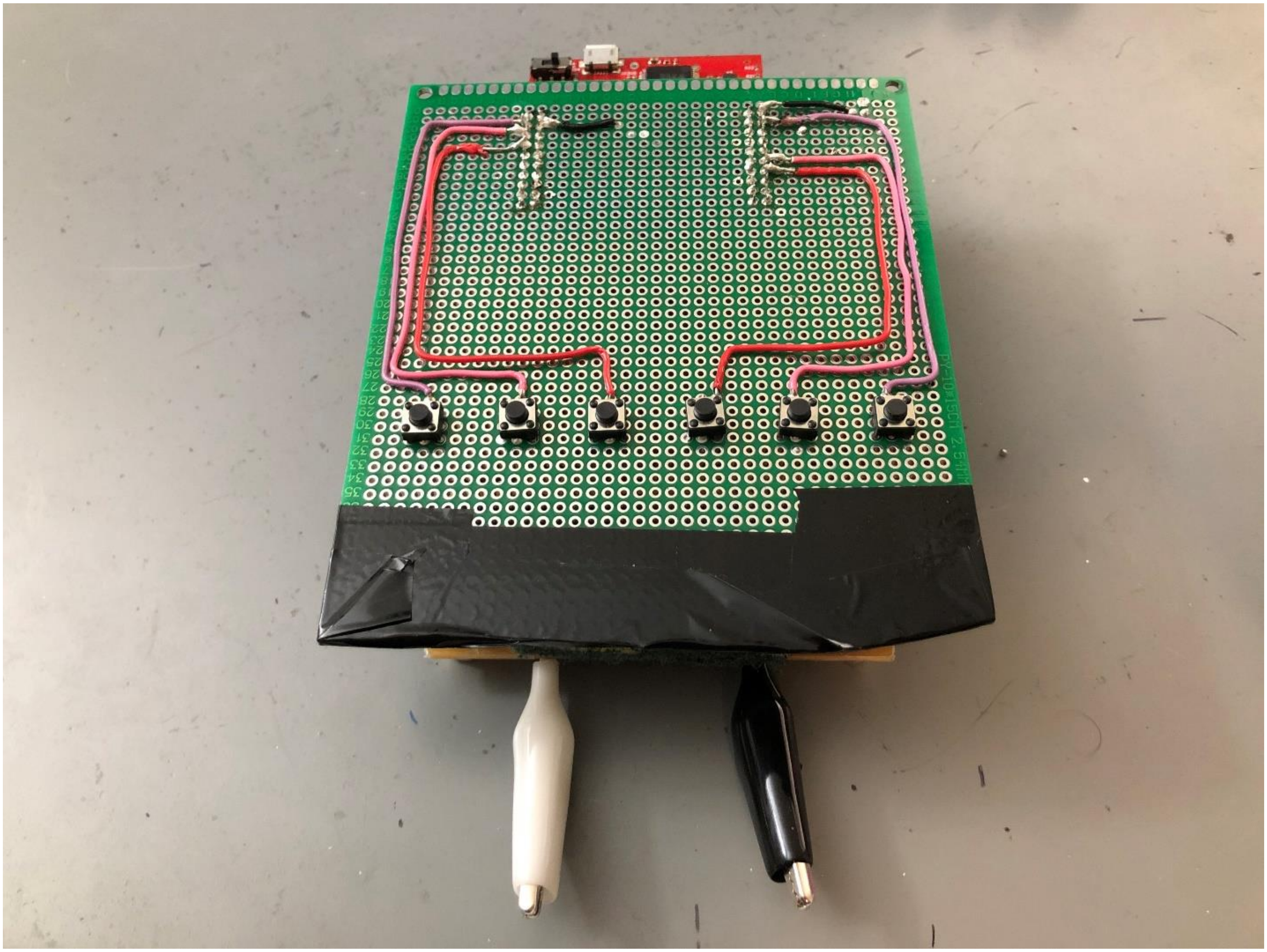


Figure 2a - Low Cost Impedance Meter

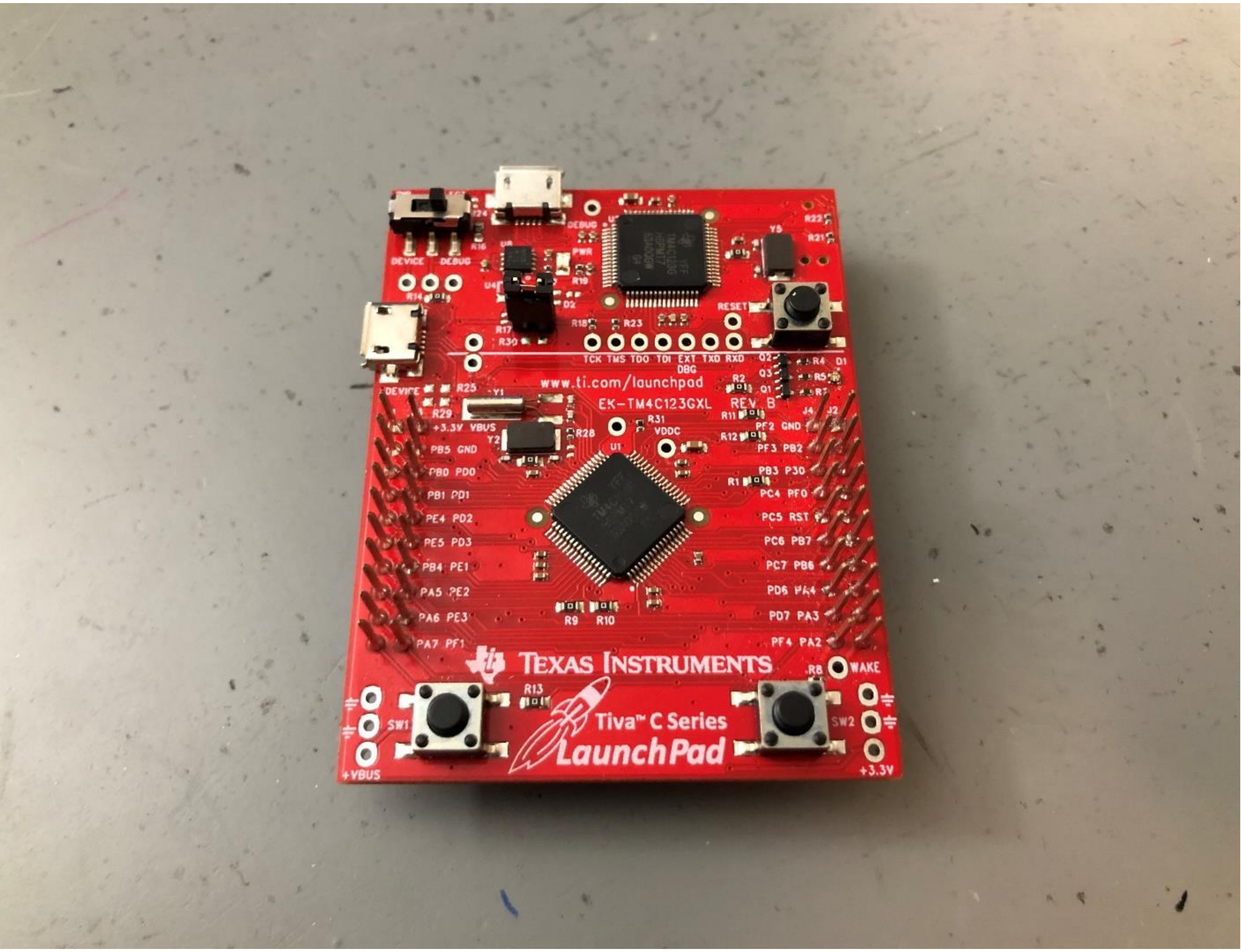


Figure 2b – TM4C123GXL Evaluation Board

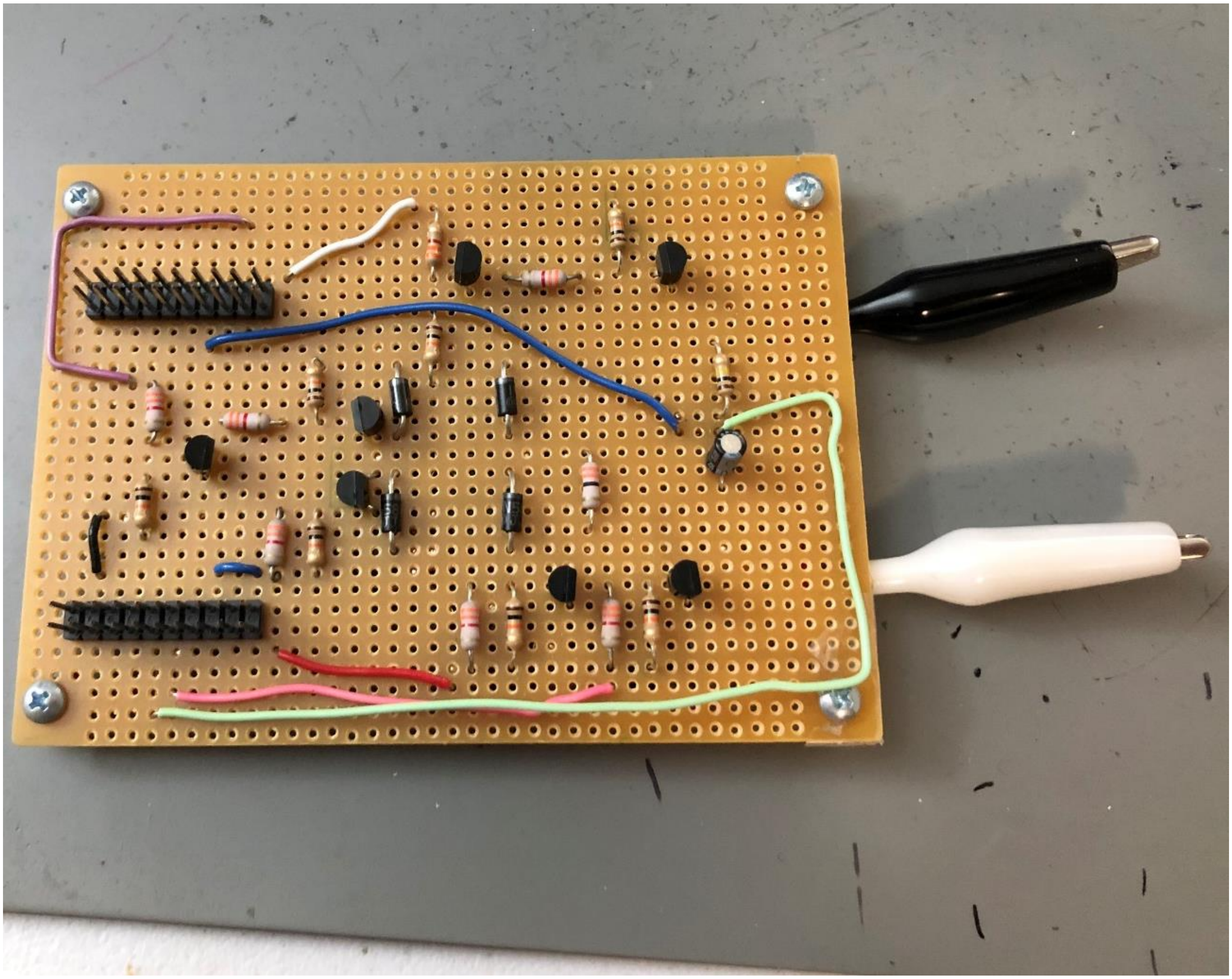


Figure 2c – Bottom board following measurements schematic

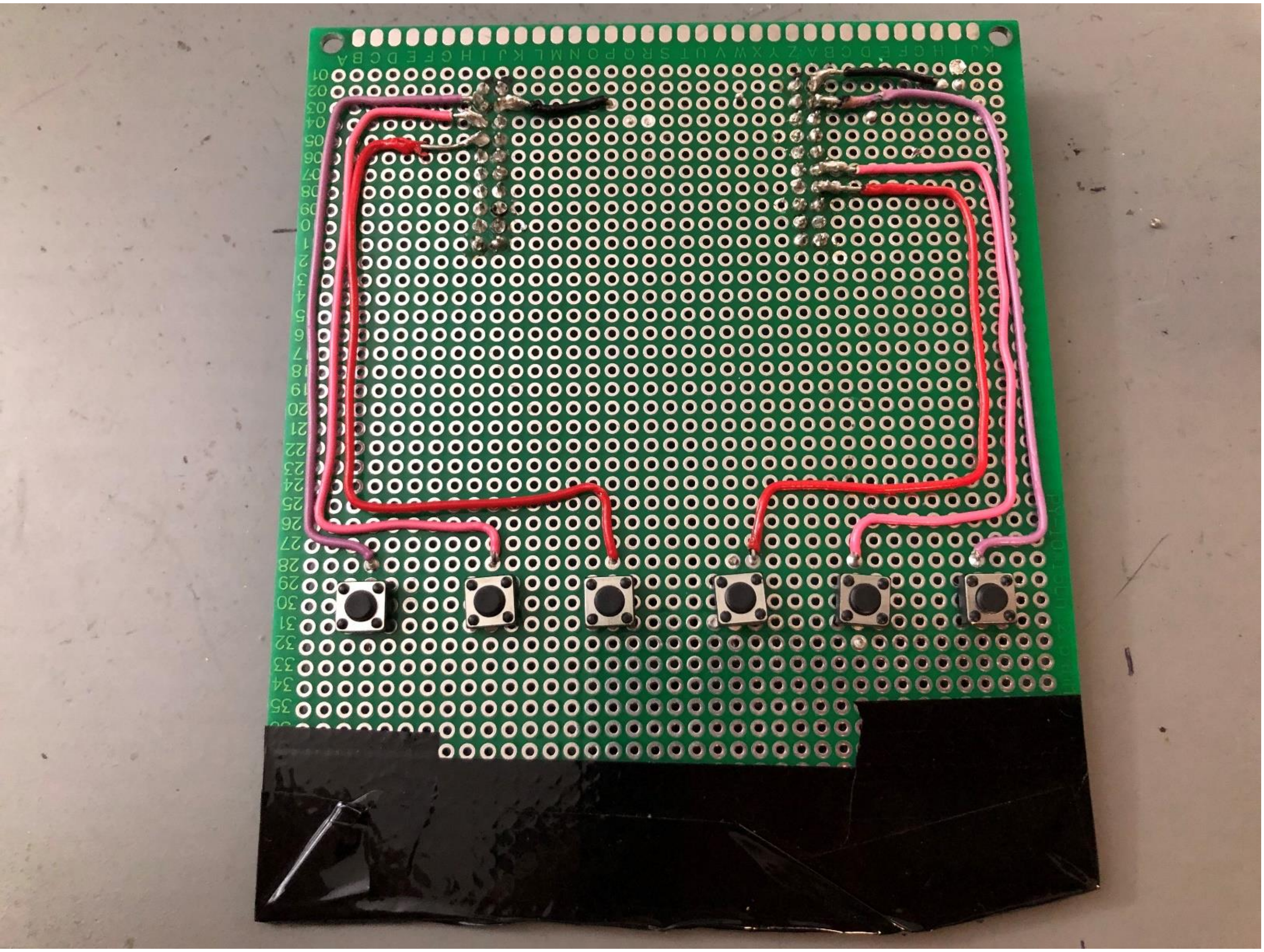


Figure 2d – Top board with Pushbuttons

Theory of Operation

This section will explain how each of the major LCR commands will operate. Resistance, Capacitance, Inductance, and ESR all use an Analog to Digital converter to read a raw analog voltage which gets converted to a digital signal and returns a value that can be used for calculation.

Measuring Resistance

To begin the measurement of resistance, all of the pins first get grounded to ensure the circuit is off. After this, a discharging of the circuit’s capacitor is commenced by turning on the INTEGRATE and LOWSIDE pin. The discharge is on for 100 milliseconds.

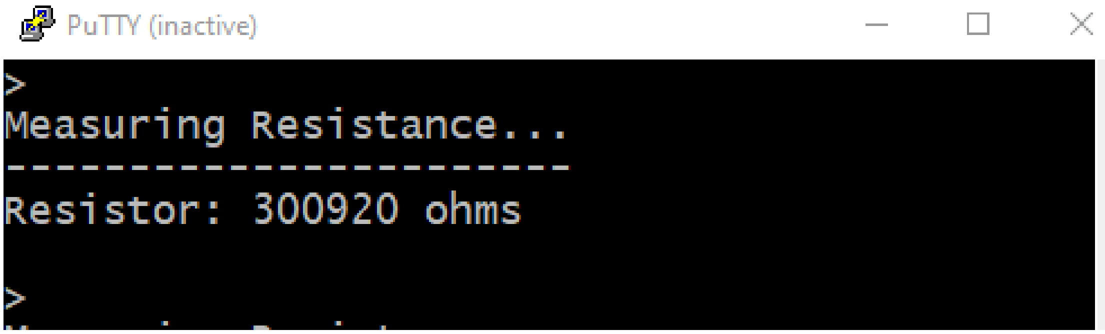
Once discharged, the timer is disabled and the TAV register is reset. Now that the timer is ready for counting, the pins for resistance measurement can be set. LOWSIDE gets set to 0 and MEASURE_LR gets set to 1, followed by turning the timer on.

Now the timer is running and the program is running through a blocking function until the ACSTAT0 register of the Comparator reaches a reference voltage of 2.469 V. If the blocking function runs for too long, there is a breakout condition for when the timer TAV register reaches a value of 0x31ABA855 to ensure that the program does not get stuck in an infinite loop. If this is triggered, resistance is returned as ‘0’.

If the reference voltage of 2.469 V is reached, the timer is turned off and the pins are grounded. There are accuracy checks for very small and very large resistance values. Once calibrated, the timer TAV register is divided by 57 to give the correct resistance value. This value is then returned as the measured resistance value.

```
// Gets
resistance
and
returns
the value
uint32_t getResistance()
{
    groundPins();                           // ground all pins first
    setPinValue(INTEGRATE, 1);
    setPinValue(LOWSIDE, 1);                // discharge
    waitMicrosecond(10e5);                  // wait for discharge


    WTIMER0_CTL_R &= ~TIMER_CTL_TAEN;       // disable timer


    WTIMER0_TAV_R = 0;                        // Reset TAV


    // Turn on pins to measure resistance
    setPinValue(LOWSIDE, 0);
    setPinValue(MEASURE_LR, 1);


    WTIMER0_CTL_R |= TIMER_CTL_TAEN;            //  Turn on timer


    // Do not commence until voltage reaches reference of 2.469V
    while (COMP_ACSTAT0_R == 0x00)
    {
        // if timer goes on too long break out
          if(WTIMER0_TAV_R > 0x31ABA855)
          {
              putsUart0("resistor took too long\t\r\n");
              WTIMER0_CTL_R &= ~TIMER_CTL_TAEN;          // Turn off counter
              groundPins();
              return(0);
          }
    }


    WTIMER0_CTL_R &= ~TIMER_CTL_TAEN;          // Turn off counter
    groundPins();                              // Ground pins
```

```
    uint32_t res = (WTIMER0_TAV_R/57) + 1;


    // If resistor is small, divide by 2
    if (res < 30 && res > 10)
    {
        res = res/2;


        setPinValue(LOWSIDE, 1);                  // Discharge again
        waitMicrosecond(10e5);                    // wait
        groundPins();
        return res;
    }


    // If resistor is between 100k and 400k, take off 3k
    if (res > 100000 && res < 400000)
    {
        res = res - 3000;


        setPinValue(LOWSIDE, 1);                  // Discharge again
        waitMicrosecond(10e5);                    // wait
        groundPins();
        return res;
    }




   //return ((WTIMER0_TAV_R/57)+1);           // Divide timer value with 57 to
get Resistance value and return
    setPinValue(LOWSIDE, 1);                  // Discharge again
    waitMicrosecond(10e5);                    // wait
    groundPins();
    return res;
}
```



Measuring Capacitance

All of the pins are grounded like the resistance function before commencing. The timer is disabled and the TAV register is reset before counting. MEASURE_C and HIGHSIDE are both set to 1 and the timer is turned on.

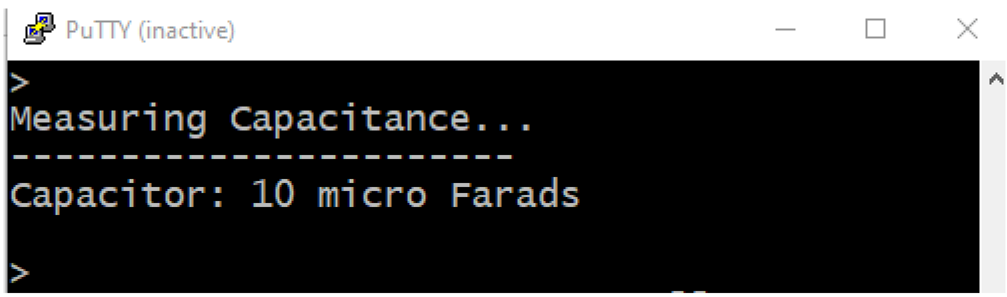
With the timer running, capacitance has the same blocking function with a condition to break out if if the TAV register has a value of 0x31ABA855. This value roughly translates to 150µ Farads or higher, this can be changed to allow for larger capacitors. If the timeout happens, the value of 0xCBAD is returned to signal that the capacitor took too long.

If the reference voltage of 2.469 V is reached, the blocking function stops and the timer is turned off. The TAV register value is multiplied by a constant of 0.00000018 to return the capacitance value in units of µ Farads.

```
// Gets
Capacitance
and returns
value
```

```
uint32_t getCapacitance()
{
    groundPins();
    //setPinValue(LOWSIDE, 1);                  // discharge
    waitMicrosecond(10e5);                    // wait for discharge


    WTIMER0_CTL_R &= ~TIMER_CTL_TAEN;        // disable timer


    WTIMER0_TAV_R = 0;                        // Reset TAV
    setPinValue(MEASURE_C, 1);
    setPinValue(LOWSIDE, 0);
    setPinValue(HIGHSIDE, 1);
    WTIMER0_CTL_R |= TIMER_CTL_TAEN;         // Turn on Timer


    // Turn on pins to measure capacitance







    // Do not commence until voltage reaches reference of 2.469V
    while (COMP_ACSTAT0_R == 0x00)
    {
        // if timer goes on too long break out (breaks @ approx 150 micro)
        if(WTIMER0_TAV_R > 0x31ABA855)
        {
            putsUart0("capacitor took too long\t\r\n");
            WTIMER0_CTL_R &= ~TIMER_CTL_TAEN;          // Turn off counter
            groundPins();


            setPinValue(LOWSIDE, 1);                  // Discharge again
            waitMicrosecond(10e5);                    // wait


            groundPins();
            return(NOT_CAP);
        }
    }
```

```
    WTIMER0_CTL_R &= ~TIMER_CTL_TAEN;          // Turn off counter
    groundPins();                              // Ground pins


    setPinValue(LOWSIDE, 1);                  // Discharge again
    waitMicrosecond(10e5);                    // wait


    groundPins();
    return ((WTIMER0_TAV_R*CAP_CONS));         // Multiply timer value with
capacitor constant and return
}
```



Measuring ESR

All pins are grounded and a LOWSIDE is set to 1 to begin discharging for 100 milliseconds. Raw voltage is then read from DUT2 and saved as a double. Ohms are calculated using the voltage divider law: `ohms = ((3.3*33.0 - voltage*33.0)/voltage);`

ESR is returned in units of ohms.

```
// Gets ESR and returns value
double getESR()
{
    groundPins();
    setPinValue(MEASURE_LR, 1);
    setPinValue(LOWSIDE, 1);         // discharge
    waitMicrosecond(10e5);           // wait for discharge


    double voltage = 0.0;
    voltage = getVoltage();          // get raw voltage on PE4


    // Calculate the ohms using voltage divider law:
    double ohms = 0.0;
    ohms = ((3.3*33.0 - voltage*33.0)/voltage);
    groundPins();
    return ohms;
}
```

```
>
Measuring ESR...
-----------------------
ESR: 4.979207 ohms

>
```

Measuring Inductance

All pins are grounded and LOWSIDE is set to 1 to begin discharging for 100 milliseconds. The timer is disabled and the TAV register reset so that it will be ready to count. MEASURE_LR is set to 1 and the timer is turned on.

With the timer running, capacitance has the same blocking function with a condition to break out if the TAV register has a value of 0x31ABA855. This value roughly translates to 150µ Henries or higher, this can be changed to allow for larger inductors. If the timeout happens, the value of 0xFBAD is returned to signal that the inductor took too long.

If the reference voltage of 2.469 V is reached, the blocking function stops and the timer is turned off. The TAV register value is used as the time constant (t) for the following formula:

```
inductance =  -(r_in * t) / (log(1- (r_in * i) / 3.3));
```

The inductance then is checked for accuracy and is calibrated according to its size. The following is returned as an inductor in units of µ Henries.

```
// Gets Inductance and returns value
uint32_t getInductance()
{
    double t = 0.0;                 // time constant
    double r_in = 0.0;              // Rin = ESR + 33 ohms
    double esr = getESR();          // esr
    double i = 0.0;                 // current
    double inductance = 0.0;        // inductance (which will be returned)


    groundPins();
    setPinValue(MEASURE_C, 1);
```

```
    setPinValue(LOWSIDE, 1);        // discharge
    waitMicrosecond(10e5);          // wait for discharge


    setPinValue(MEASURE_C, 0);


    WTIMER0_CTL_R &= ~TIMER_CTL_TAEN;          // Disable Timer
    WTIMER0_TAV_R = 0;                        // Reset TAV


    WTIMER0_CTL_R |= TIMER_CTL_TAEN;        // Turn on timer


    setPinValue(MEASURE_LR, 1);              // Turn on pin to measure
inductance


    // Do not commence until voltage reaches reference of 2.469V
      while (COMP_ACSTAT0_R == 0x00)
      {
          // if timer goes on too long break out (breaks @ approx 150 micro)
          if(WTIMER0_TAV_R > 0x31ABA855)
          {
              putsUart0("inductor took too long\t\r\n");
              WTIMER0_CTL_R &= ~TIMER_CTL_TAEN;          // Turn off counter
              groundPins();
              return(NOT_IND);
          }
      }




    WTIMER0_CTL_R &= ~TIMER_CTL_TAEN;          // Turn off counter


    t = ((double) (WTIMER0_TAV_R)) / (40000000);    // time constant = timer
value / sys clock
    r_in = esr + 33.0;                              // Rin value
    i = V_REF / r_in;                               // 2.469V / Rin to get
current
```

```
            inductance =  -(r_in * t) / (log(1- (r_in * i) / 3.3));
            float ind;




            // Smaller value checks for accuracy:
            if(inductance < 0.0001 && inductance >= 0.00005035)
            {
                ind = inductance/2;
                groundPins();
                return (double) (ind*1e6);
            }
            // If under 25 uH or so, divide again
            if(inductance < 0.00005035)
            {
                ind = inductance/4;
                groundPins();
                return (double) (ind*1e6);
            }


            groundPins();
            return (double) (inductance*1e6);
        }
```

Auto Measurement

When the auto command is commenced, the device immediately measures Resistance, Capacitance, and Inductance of the device under test. Below are the following test cases specific to this device for determining what type of component is connected:

<u>Resistor</u>

If the component is a resistor, the capacitance should have returned 0xCBAD and the inductance should have returned 0xFBAD. If these values are received, then the device will print the resistance returned.

<u>Capacitor</u>

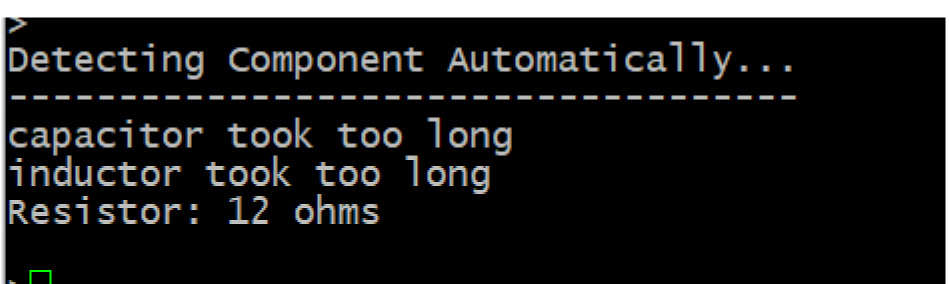
If the component is a capacitor, the inductance returned should be really large and the resistance very small. If both are true, then the returned capacitance value is printed.

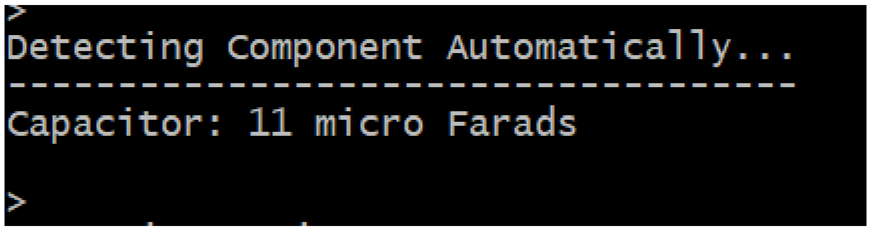
<u>Inductor</u>

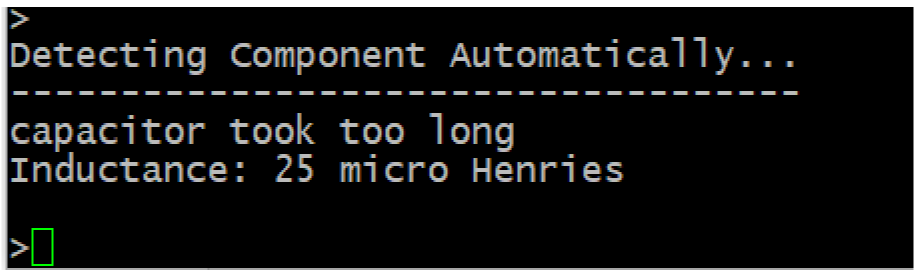
If the component is an inductor, the capacitor will return 0xCBAD and the resistance will be fairly small. If both are true, then the returned inductance value is printed.

```
// Function to
get
measurement
automatically.
This function
will determine
if the
connected
component is
// either a Resistor, Capacitor, or an Inductor
void auto_measure()
{
    uint32_t res = getResistance();   // 0 = NOT RES value
    waitMicrosecond(100000);          // put some delay to avoid
interference
    uint32_t cap = getCapacitance();  // 0xCBAD = NOT CAP value
    waitMicrosecond(100000);          // put some delay to avoid
interference
    uint32_t ind = getInductance();   //  0xFBAD = NOT IND value


    // if NOT cap and NOT ind, print resistance
```

```
        if(cap == NOT_CAP && ind == NOT_IND)
        {
            char res_str[20];
            putsUart0("Resistor: ");
            sprintf(res_str,"%d",res);
            putsUart0(res_str);
            putsUart0(" ohms");
            return;
        }
        // print cap
        if(ind > 200 && res < 10)
        {
            char cap_str[20];
            putsUart0("Capacitor: ");
            sprintf(cap_str, "%d", cap);
            putsUart0(cap_str);
            putsUart0(" micro Farads");
            return;
        }
        // print ind
        if(cap == NOT_CAP && res < 100)
        {
            char ind_str[150];
            putsUart0("Inductance: ");
            sprintf(ind_str, "%d", ind);
            putsUart0(ind_str);
            putsUart0(" micro Henries");
            return;
        }


    }
```







Conclusion

After building the hardware and integrating the software, a fully functional Low Cost Impedance Meter was successfully assembled and can be used to measure Resistance, Capacitance, Inductance, and ESR of a device under test. With more calibration and improved quality of parts and circuity, this Impedance Meter can be expanded upon for better accuracy.