New York City College of Technology

Department of Electrical & Telecommunications Engineering Technology

EET 2271 – OL51 - Circuit Analysis Laboratory

Fall 2022 Semester

Project 2: Designing of an Ohmmeter

Nicholas Pillay, Amarjot Singh, Jason Alarcon Ali, Devesh Ramsingh

Due Date: 11/19/2022

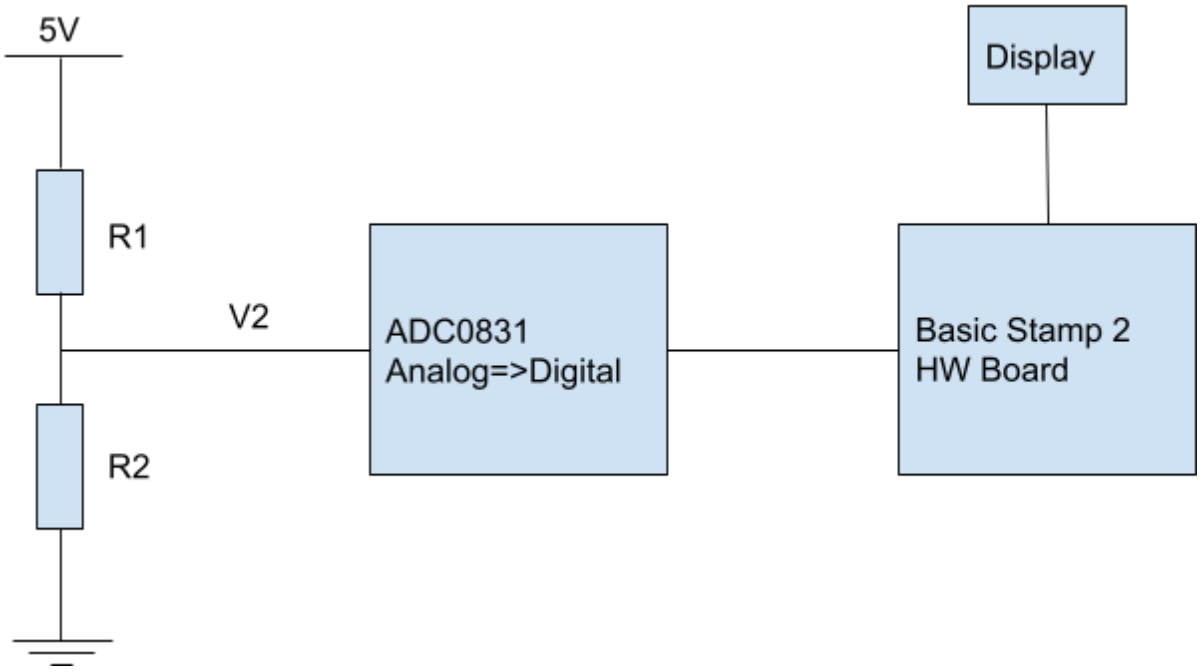
Instructor: Professor Aale Naqvi

Table of Contents

Objective

The goal of the project is to design an Ohmmeter by using the Basic Stamp. The output of the Ohmmeter is the computer monitor and so the value of a resistor that you want to measure must be displayed on the monitor. A switch is also incorporated into the design for determining the mode of operation. If the switch is turned OFF the output should display "Turn on the switch for Ohmmeter Mode". However, when a resistor is placed in the socket for it to be measured and the switch is turned ON the Ohmmeter measures the resistor and displays "Resistance is ohms".

Theory of Operation



For this lab we will be using the Voltage Division circuit to calculate R2. The Voltage Division circuit's analog outputs will be bridged by the ADC0831 to the Basic Stamp 2 HW Board for computation and then the output will be displayed on the Display.

$$\frac{V_2}{V_1} = \frac{R_2}{R_1}$$

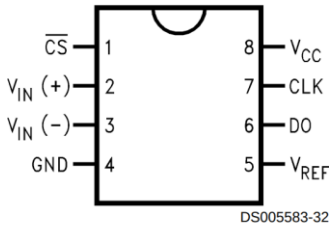
$$\frac{R_1 V_2}{V_1} = R_2$$

$$\frac{R_1 V_2}{5V - V_2} = R_2$$

The way we calculate the R2 is we take the V2 voltage. The ratio of V2 to V1 should be equal to the ratio of R2 to R1, where V1 is the voltage across R1. R1 is constant, we choose that

resistance. V1 is the same thing as 5V-V2 because of Kirchhoff's Voltage Law. Therefore, we have our equation (V2/V1)(R1)=R2, which, when expanded is R1*[V2/(5V-V2)]=R2.

ADC0831 Single Differential Input Dual-In-Line Package (N)



Top View

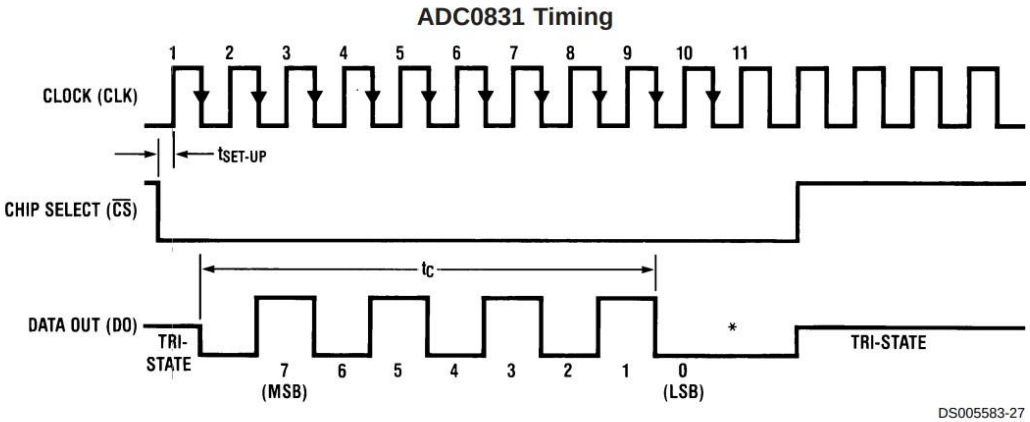
The ADC0831 converts the analog into 8 bit digital. The voltage supply, $V_{CC,}$ can range between 4.5 and 6.5 V_{DC}. The voltage range to the pins for all analog and digital inputs is from -0.3V to V_{CC}+0.3V.

The voltage we are measuring, V2, goes into the V_{IN} (+) pin and it is compared to the V_{REF} input, which in this case is the same as the V_{CC}, the 5V. That measurement is converted into a 8 bit digital signal.

The CS' is the chip select, it turns the chip on and off so that the digital output, DO, can be given according to the CLK rate. For our Basic Stamp 2 board we want 8 bits, 0-255 states, and we want the clock rate to be 100ms.

Timing Diagrams (Continued)

ADC0831 Timing

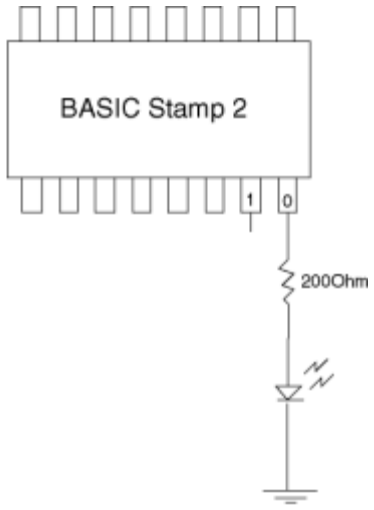


*LSB first output not available on ADC0831.

We will expect more accurate results when R1 is closer to R2, because the 8 bit resolution makes it hard to measure exact quantities. That error ends up being very large after calculations when V_2 is very close to the V_{REF} or 0V. For example, if V2 is 0.009V, the ADC0831 might measure it as $0000\ 0001_2$, 0.0195V; after calculations that makes the ratio of V2:V1=0.003915, instead of 0.001803, which is a large error.

Pre-lab

1) LED Blink on Output 0




```
Software:

TOP:

        High 0  ' turn on the LED
        PAUSE 500       ' pause for 500 msecs

        Low 0           ' turn off the LED
        PAUSE 500

        GOTO TOP
```

There is a wire from P0 to a 200 Ω resistor, which is connected to an LED then ground. The code makes it so the LED turns on for 500 ms, then turns it off for 500 ms and that repeats.

2) LED Blink on Output 0 using another method

```
        DELAY_TIME CON 500

TOP:

        HIGH 0          ' turn on the LED

        PAUSE DELAY_TIME' pause for 500 msecs

        LOW 0           ' turn off the LED
        PAUSE DELAY_TIME

        GOTO TOP
```

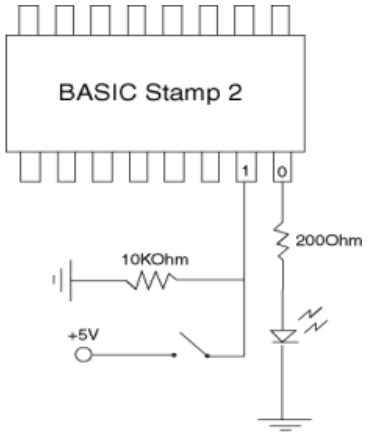
The wiring diagram is the same, but the code sets a variable called DELAY_TIME to a constant value (which we put as 500 ms) and that is the time the LED stays on and the time it stays off.

3) LED Blink on Output 0 50 times using Subroutine

```
TIMES   VAR BYTE
LED     CON 0   ' LED is on pin 0
DELAY_TIME CON 500


TOP:

        FOR TIMES = 1 TO 50
           GOSUB FLASH
        NEXT

        DEBUG "Done! "

DONE:
Debug "here"
        GOTO DONE


FLASH:          ' winks LED on and off one time

        HIGH LED                ' turn on the LED
        PAUSE DELAY_TIME        ' pause for 500 msecs

        TOGGLE LED              ' turn off the LED



         PAUSE DELAY_TIME
         DEBUG? TIMES
         RETURN
```

The TOP routine goes to the FLASH subroutine to blink on and off for a set delay time (500 ms) 50 times and then goes to the DONE routine, which shows a string saying DONE!.

5) Reading a state of a switch



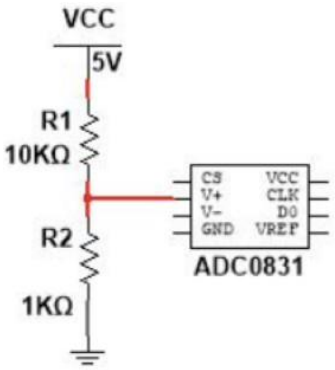
Code

```
infiniteLoop: 'label

'if button is pressed, goto subroutine Flash
IF IN1 = 1 THEN Flash

GOTO infiniteLoop 'loop

Flash: 'subroutine labeled Flash

HIGH 0 'turn on LED
PAUSE 1000 'wait 1 second
LOW 0 'turn off LED
PAUSE 1000 'wait 1 second

RETURN 'go back to the rest of the program
```

P1 is 1 if the switch is on and 0 if the switch is off. If it's 0 it is caught in a loop until the switch is on, and if it’s 1 it goes to the FLASH subroutine and blinks the LED for 1 second on and off.

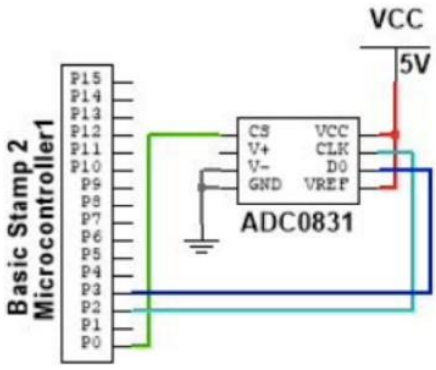
Preparation

To begin building our circuit, we must first get the components necessary for the project. We purchased the different resistor values used for the ohmmeter, the board and LCD from Parallax, the ADC0831 chip, the Basic Stamp software, and a switch. Once we have all the parts, we must wire the microcontroller to the board and LCD correctly so that we can get the correct result. For the ADC0831 to work as intended, we have to make sure that the proper voltages go to the pins so that the chip gives us the correct reading, To start, V+ needs to have the voltage of R2 that will be compared to the reference voltage, so a wire needs to go from between the two resistors to Pin 2 as shown in the picture below.



$$V_{R2} = V_{CC}\left(\frac{R_2}{R_1+R_2}\right)$$

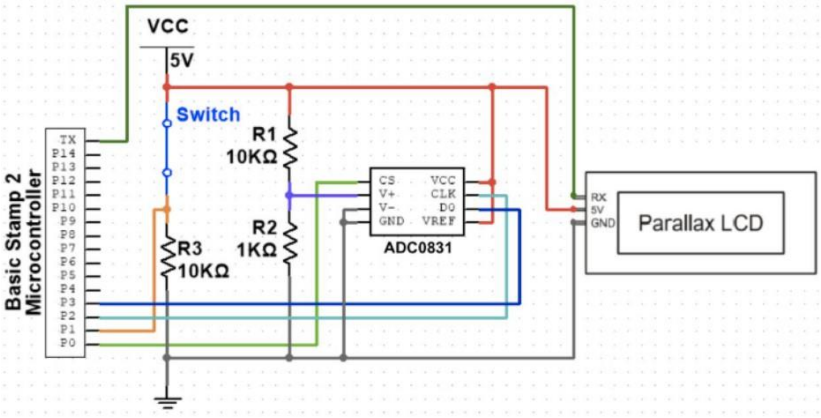
After that the 5V needs to connect to the chip to give it power and to V_{REF} as well. The clock signal then needs to be sent to the clock from the microcontroller and D_O is the data output from the chip which is sent to the microcontroller. The connection can be seen in the image below.



$$D_O = \left(\frac{255}{V_{REF}}\right)V_{R2}$$

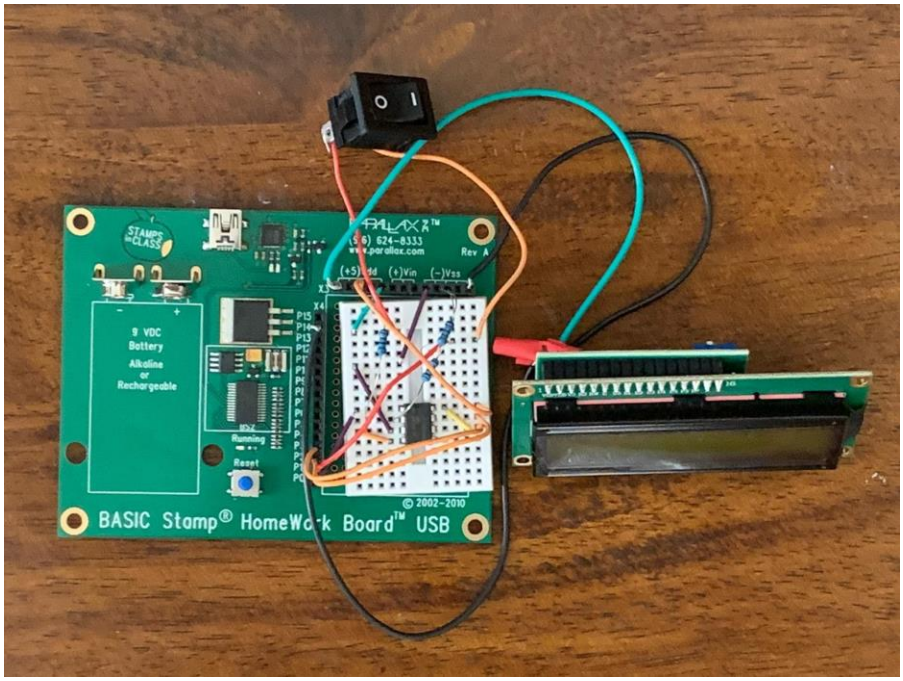
After that the chip select needs to be connected to the microcontroller and the V- and ground pins need to be connected to ground. Then the switch is connected to the 5V and ground,

with a wire after the switch connecting to the microcontroller detecting the position of the switch. Lastly is the LCD connected to the 5V, ground, and a socket in the microcontroller that acts as the TX pin. All of these connections are then carried out and combined, resulting in a diagram that can be seen below.

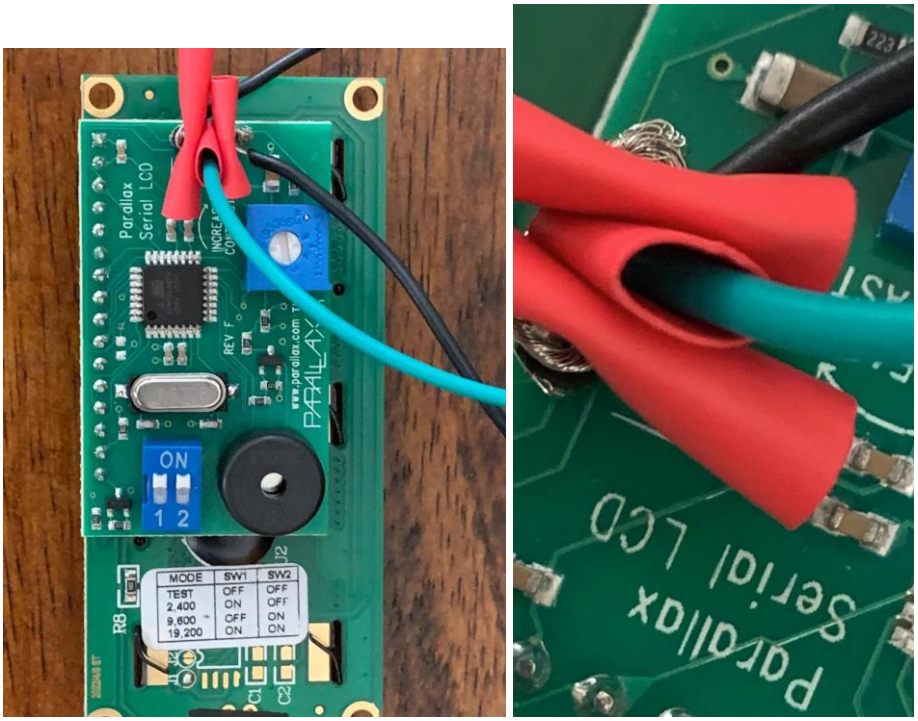


Manufacturing and Assembly Procedure

We wired the circuit according to the layout above, the only difference being that the TX pin was chosen to be P14 instead of P15. The only issue with the LCD connections was that the wire we were using for the connections on the board did not seem to work for the LCD, as it would not light up. After troubleshooting we found the problem to be fixed by using 24 gauge wire we had on hand and the LCD lit up accompanied by a start-up sound. The finished circuit looked like the picture below.



It also took some ingenuity to keep the wires from touching each other. The quick solution we came up with was to use dome heat shrink tubes to completely isolate the middle pin and squeeze more tubes between the pins to make sure the wires do not fall out. Pictures of the back of the LCD show that it was not the most elegant solution, but it worked.



With the circuit complete we must work on the code so that the microcontroller works as an ohmmeter. That involved making sure the different points that we connect to are associated with the correct variable, so the CS is at P0, switch position detector at P1, D0 at P2, clock at P3, and the TX pin for the LCD at P14. The baud rate, as well as different variables and constants that are used for the computation are defined at the start so that we can work it as the program goes through and obtains data from the chip. So first, the program checks the switch position in the INFINITELOOP function, which either proceeds to the MAIN function if the switch is on, or the LCDOFF function if the switch is off. Going to LCDOFF, the LCD shows text saying, “Turn on the switch” and pauses for 100 ms, going back to the INFINITELOOP function until the switch is flipped on.

When the switch is on and we go to the MAIN function, the CS pin is high, an Arbitrary variable is given a value for the clock pulse manually (100 in our case), and goes to the ADC0831 function. After that the CS is low, a clock pulse is pulsed into the chip with our arbitrary value and a digital output ADRESULT is produced from D0, having CS go back to high and returning to the MAIN function.

In the MAIN function again, a value for R2 is generated through several steps; multiplying ADRESULT and R1, defining B as the result of 255 - ADRESULT, dividing the first instance of R2 by B, and multiplying that by 100. We then wanted to see R2 and ADRESULT in the window when running the program to see what the values are for troubleshooting purposes, and then proceed to the LCDON function.

In the LCDON function we display the string saying "Resistance is ", followed by the value of R2, and then a string saying " ohms". This is held for 2 seconds, then returns to the MAIN function, which in turn returns to the INFINITELOOP function. The code that we used as the result of this can be seen below.

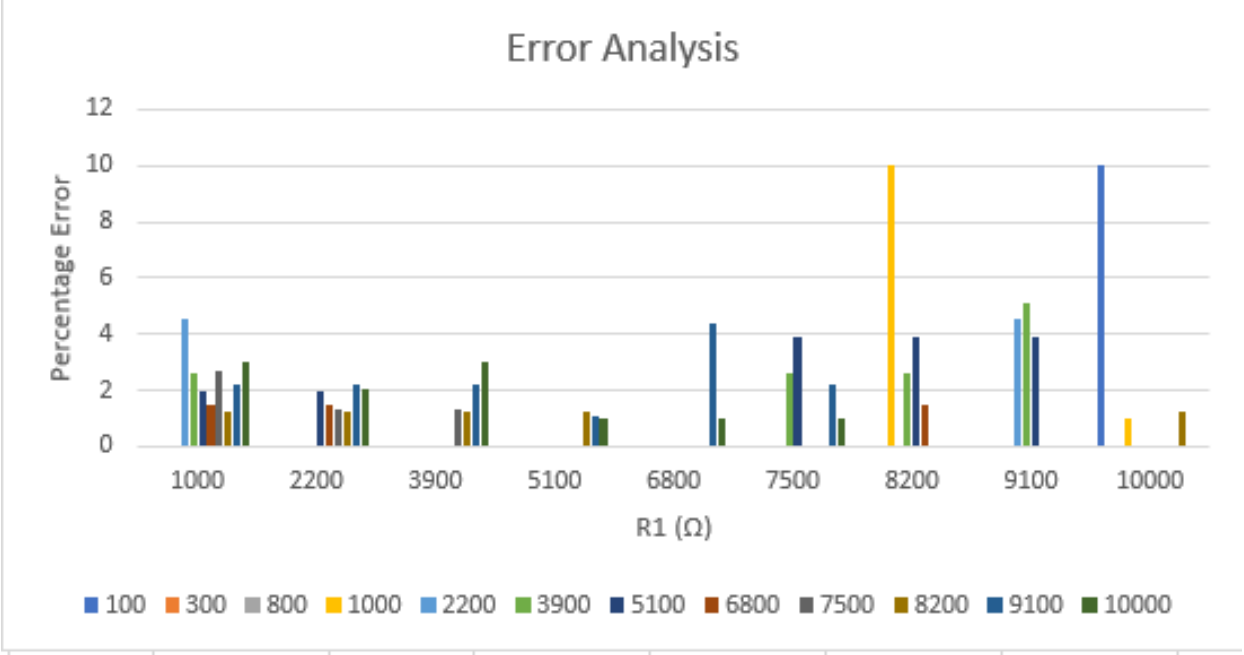
```
' {$STAMP BS2}
' {$PBASIC 2.5}
' {$PORT COM3}

CS CON 0  'CHIP SELECT PIN.
D0 CON 3  'DATA PIN.
CLK CON 2  'CLOCK PIN.

Arbitrary VAR Byte    'Defines Variable of the size Byte.
ADRESULT VAR Byte
R2 VAR Word
R1 CON 100      'Defines R1 as constant.
B VAR Word

TXPIN CON 14
BAUD19200 CON 32

INFINITELOOP:      'Reads switch 1=MAIN 0=LCDOFF
  IF IN1 = 1 THEN MAIN
  GOTO LCDOFF

MAIN:          'Calc R2 from re
  HIGH CS
  Arbitrary = 100
  GOSUB ADC0831    'Runs ADC part
  R2 = ADRESULT*R1
  B = 255 - ADRESULT
  R2 = R2 / B
  R2 = 100*R2
  DEBUG? R2
  DEBUG? ADRESULT
  GOSUB LCDON    'Runs LCDON part
  RETURN      'Goes back TO infinite loop
```

```
ADC0831:        'Creates CS pulse, sets clock rate TO 100ms FOR chip
  LOW CS
  PULSIN CLK, 0, Arbitrary
  SHIFTIN DO, CLK, MSBPOST, [ADRESULT \ 9]
  HIGH CS
  RETURN      'Returns TO main

LCDOFF:                 'Tells you turn it on in loop.
  SEROUT Txpin, Baud19200, [12,17]
  SEROUT Txpin, Baud19200, ["Turn on the switch", 13]
  PAUSE 100
  SEROUT Txpin, Baud19200, [12]
  SEROUT Txpin, Baud19200, [18]
  GOTO INFINITELOOP

LCDON:            'Returns the OUTPUT on the display from main
  SEROUT Txpin, Baud19200,  [12,17]
  SEROUT Txpin, Baud19200 ,["Resistance is   ", DEC R2," ohms", 13]
  PAUSE 2000
  SEROUT Txpin, Baud19200, [12]
  SEROUT Txpin, Baud19200, [18]
  RETURN
```

With the software and hardware done, all that was left was to load the microcontroller with the program and to run it, seeing if it works. When we put a 10 kΩ resistor for R1 and 100 Ω resistor for R2 to test, we got a result on the LCD saying that the resistance is 110 ohms. Seeing that the ohmmeter works, we replaced R2 and worked our way up from 100 Ω to 300 Ω, 820 Ω, 1 kΩ, 2.2 kΩ, 3.9 kΩ, 5.1 kΩ, 6.8 kΩ, 7.5 kΩ, 8.2 kΩ, 9.1 kΩ, and then 10 kΩ, recording the resistance value we got from the LCD for each R2. We then worked our way down replacing the R1 resistor to 9.1 kΩ and repeated the steps of recording the different R2 values, down to 8.2 kΩ, 7.5 kΩ, 6.8 kΩ, 5.1 kΩ, 3.9 kΩ, 2.2 kΩ, and 1 kΩ. With all those values we created tables and graphs in Excel showing the different R2 values obtained as well as the percentage error for every point we recorded.

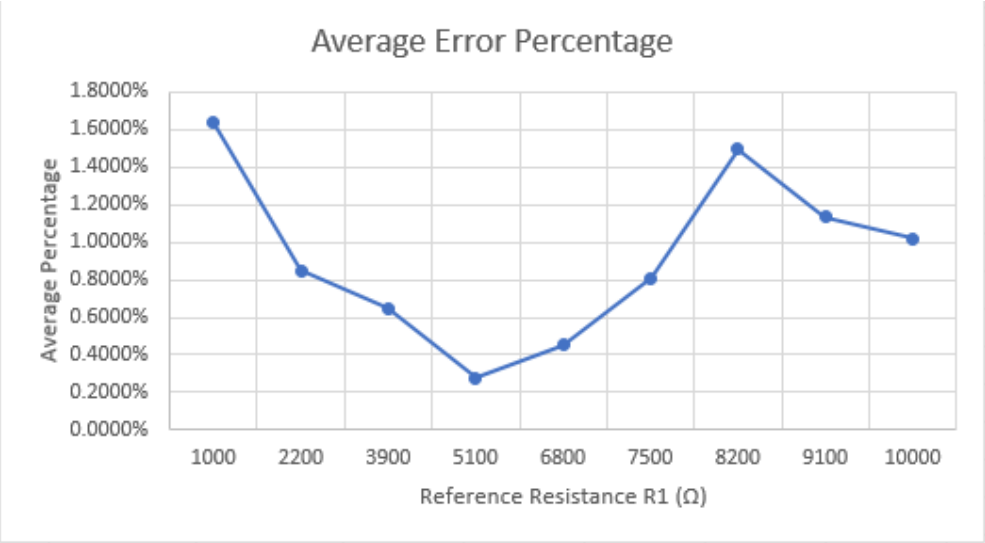
	1000	2200	3900	5100	6800	7500	8200	9100	10000
100	100	100	100	100	100	100	100	100	110
300	300	300	300	300	300	300	300	300	300
800	800	800	800	800	800	800	800	800	800
1000	1000	1000	1000	1000	1000	1000	1100	1000	990
2200	2300	2200	2200	2200	2200	2200	2200	2300	2200
3900	4000	3900	3900	3900	3900	3800	3800	4100	3900
5100	5200	5200	5100	5100	5100	5300	4900	5300	5100
6800	6900	6900	6800	6800	6800	6800	6700	6800	6800
7500	7700	7600	7400	7500	7500	7500	7500	7500	7500
8200	8300	8300	8100	8100	8200	8200	8200	8200	8300
9100	9300	9300	9300	9200	8700	8900	9100	9100	9100
10000	9700	9800	10300	9900	9900	9900	10000	10000	10000

	1000	2200	3900	5100	6800	7500	8200	9100	10000
100	0	0	0	0	0	0	0	0	10
300	0	0	0	0	0	0	0	0	0
800	0	0	0	0	0	0	0	0	0
1000	0	0	0	0	0	0	10	0	1
2200	4.54545455	0	0	0	0	0	0	4.54545455	0
3900	2.56410256	0	0	0	0	2.56410256	2.56410256	5.12820513	0
5100	1.96078431	1.960784	0	0	0	3.92156863	3.92156863	3.92156863	0
6800	1.47058824	1.470588	0	0	0	0	1.47058824	0	0
7500	2.66666667	1.333333	1.33333333	0	0	0	0	0	0
8200	1.2195122	1.219512	1.2195122	1.2195122	0	0	0	0	1.2195122
9100	2.1978022	2.197802	2.1978022	1.0989011	4.3956044	2.1978022	0	0	0
10000	3	2	3	1	1	1	0	0	0



We also plotted the average percentage error for each R1 and obtained this graph.

1000	2200	3900	5100	6800	7500	8200	9100	10000
1.6354%	0.8485%	0.6459%	0.2765%	0.4496%	0.8070%	1.4964%	1.1329%	1.0183%



The errors can be attributed to many factors, the main one being the resistance of the resistor not being the exact value due to there being a tolerance.

Conclusion and Comment

Therefore, this experiment has taught us a lot about designing a ohmmeter using a Basic Stamp 2 Microcontroller and Basic Stamp Software. One difficulty we ran into when performing this experiment was with the connections on the microcontroller. The hook-up wire we were using on the board worked well for everything except the LCD, which would not turn on when. We then used a 24-gauge wire for the 3 LCD connections to the TX pin, 5V, and ground, and found that the problem was solved. We still needed a way to secure the wires so that they would not touch the other pins or fall off, so we used heat shrink tubes to keep the wires isolated and tight against the tubes so that they would not move. Our progress with this experiment was that we successfully built a working ohmmeter which taught us the basics of combining hardware and software for various designs. The reason we know the circuit is working is because, when we connected a resistor that we knew the resistance of, our ohmmeter read us the correct resistance (with little percent error). Lastly the display also read everything that we programmed it to show.

References

- Provided File Projects for EET2271 pdf
- Provided Electronic Handbook pdf
- ADC0831CCN - ADC0831 8-bit A/D converter datasheet. Futurlec. (n.d.). https://www.futurlec.com/ADConv/adc0831a.shtml