IRIS NICOLE CARSON 4 BS CpE March 21, 2023
ANTONIO RAFAEL CASTRO
JOSHUA MANUEL LOUISE KEMPIS

Project 2 Final Report **EARLY SUBMISSION DRAFT**

Interfacing a Temperature Sensor

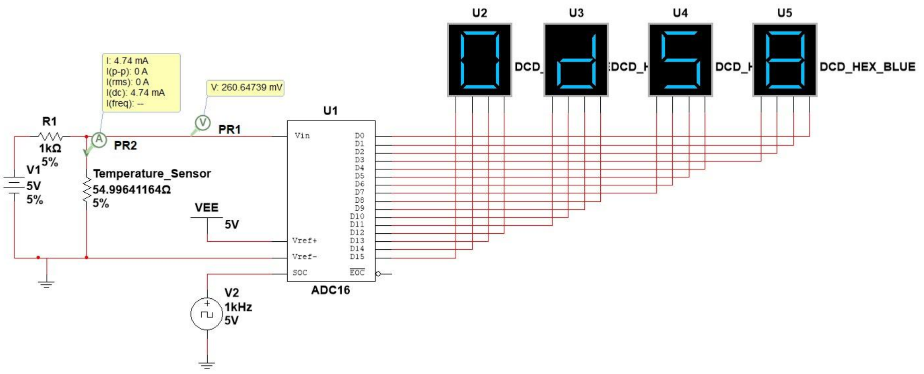
Overview

A program is created to simulate how a microprocessor system determines temperature values picked up from a temperature sensor. The program utilizes an analog to digital converter which detects the initial input from a given temperature stimulus in the form of a resistance value and passes it onto an equation that solves the output temperature according to what was read from the ADC output. This output is reflected into

Code Collaboration Overview

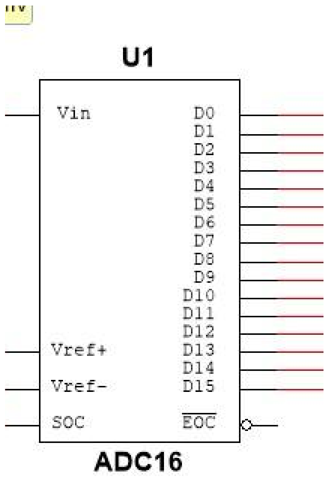
The code is collaborated between three members which contribute to the common GitHub repository. The repository contains all the code and drafts that contribute to the final output. Using the GitHub platform, contributions and updates can easily be logged and monitored as well as resolving revision conflicts on the same code.

Temp Sensor Circuit Overview



A circuit was constructed to display how the temperature sensor would be interfaced together with the ADC on how the measured temperature is outputted through a series of seven-segment display representing 16 bit values. Resistor R1 represents a 1k Ohm resistor in parallel to the measured resistance given off by the temperature sensor.

These inputs are fed into the U1 circuit component that handles the processing of the measured data into its respective output.



The ADC takes an analog signal input in its SOC pin. In this case, V2 is a square wave with a value of 5V and a frequency of 1 kHz. Pins D0 to D15 represents the digital output pins to where every 4 pins is connected to its respective display output.

The Vref+ and Vref- acts as the VCC and GND pin equivalents in the circuit. The Vin pin represents the voltage input across the temp sensor represented by the variable resistor in the circuit.

For this virtual temperature sensor, the range of values maps between -45°C to 120°C. In this manner, the output values can be considered within the range of 20°C increments. This would then provide a range of temperature values that would be tested in the circuit.

These values can be represented in Celsius:

```
{-45, -25, -5, 15, 35, 55, 75, 90, 105, 120};
```

Or in their equivalent values that can be read by the circuit output

```
{"F9B8", "EB11", "CB00", "9A48", "6761","400F", "26C0", "1ABE", "12B9", "0D58"};
```

The circuit was modified as seen in **Appendix A**, to consider the values of each temperature as a parallel resistor input that can be controlled via switch. This would help in the testing process as the resistance values represented by each temperature value in the range can easily be selected.

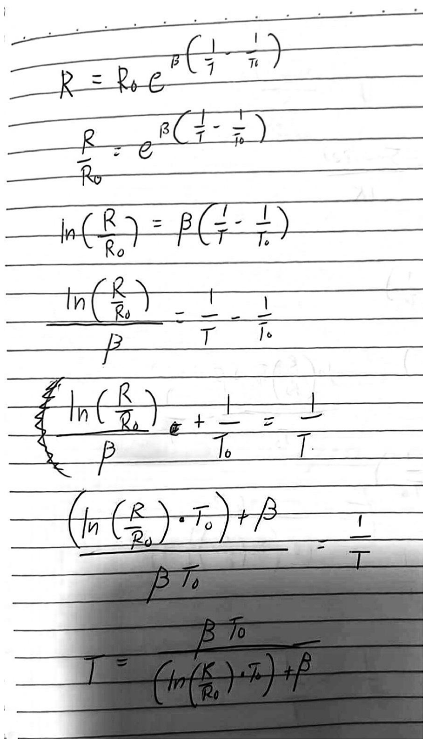
Creating the Resistor Process Implementation

The equation:

$$R(T) = R_0 e^{\beta\left(\frac{1}{T} - \frac{1}{T_0}\right)}$$

Was used to obtain the value of R(T) given the different initial circumstances that the virtual Temp sensor may receive. The equation follows several constant values that can be derived by computing how they factor into the main equation. The value for R_0 and β are given with the equations:

$$\beta = \frac{\ln 3}{\frac{1}{273.15} - \frac{1}{298.15}} \qquad R_0 = \frac{3000}{e^{\beta\left(\frac{1}{273.15} - \frac{1}{T_0}\right)}}$$



$$R = R_0 e^{\beta\left(\frac{1}{T} - \frac{1}{T_0}\right)}$$

$$\frac{R}{R_0} = e^{\beta\left(\frac{1}{T} - \frac{1}{T_0}\right)}$$

$$\ln\left(\frac{R}{R_0}\right) = \beta\left(\frac{1}{T} - \frac{1}{T_0}\right)$$

$$\frac{\ln\left(\frac{R}{R_0}\right)}{\beta} = \frac{1}{T} - \frac{1}{T_0}$$

$$\frac{\ln\left(\frac{R}{R_0}\right)}{\beta} \cancel{e} + \frac{1}{T_0} = \frac{1}{T}$$

$$\frac{\left(\ln\left(\frac{R}{R_0}\right) \cdot T_0\right) + \beta}{\beta T_0} = \frac{1}{T}$$

$$T = \frac{\beta T_0}{\left(\ln\left(\frac{R}{R_0}\right) \cdot T_0\right) + \beta}$$

Furthermore, the value of T in Kelvin can be obtained through the following derivation of the equation by isolating T. For the activity, the values in Kelvin [K] can then be converted into degrees Celsius [°C].

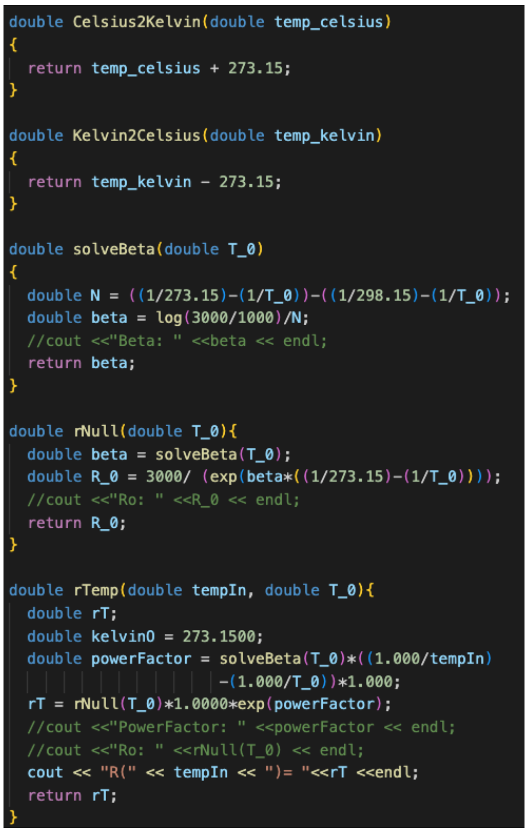
This would provide the known relationship between the values of T, R, R_0, and T_0 in the equation.

Creating the Temp Sensor Process Implementation

The C++ implementation of the Analog to Digital converter was created with corresponding values displayed in the terminal. Values are also outputted as a way to debug and check if the program is running properly when compared to the values that were manually computed in an external spreadsheet.

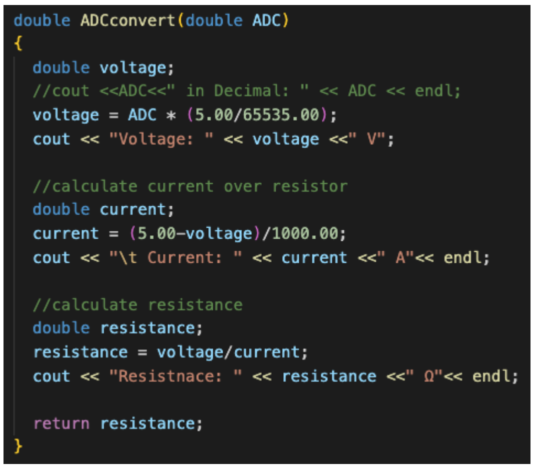
Given a set of temperature values as a stimulus for the temp sensor to respond to, it is possible to provide the necessary values to solve for the values of T, R, R_0 and T_0.

C++ Functions and Formulas



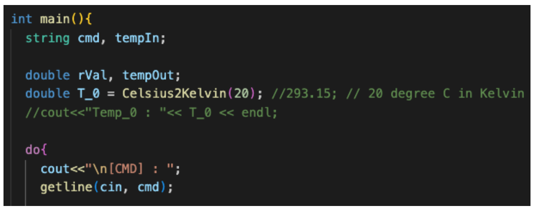
```cpp
double Celsius2Kelvin(double temp_celsius)
{
  return temp_celsius + 273.15;
}

double Kelvin2Celsius(double temp_kelvin)
{
  return temp_kelvin - 273.15;
}

double solveBeta(double T_0)
{
  double N = ((1/273.15)-(1/T_0))-((1/298.15)-(1/T_0));
  double beta = log(3000/1000)/N;
  //cout <<"Beta: " <<beta << endl;
  return beta;
}

double rNull(double T_0){
  double beta = solveBeta(T_0);
  double R_0 = 3000/ (exp(beta*((1/273.15)-(1/T_0))));
  //cout <<"Ro: " <<R_0 << endl;
  return R_0;
}

double rTemp(double tempIn, double T_0){
  double rT;
  double kelvinO = 273.1500;
  double powerFactor = solveBeta(T_0)*((1.000/tempIn)
                        -(1.000/T_0))*1.000;
  rT = rNull(T_0)*1.0000*exp(powerFactor);
  //cout <<"PowerFactor: " <<powerFactor << endl;
  //cout <<"Ro: " <<rNull(T_0) << endl;
  cout << "R(" << tempIn << ")= "<<rT <<endl;
  return rT;
}
```

Several intermediary functions were created to solve for the values using the same method as used in the spreadsheet formulas. These include the formula derived to get the value of T and the resistance value R(T).



```cpp
double ADCconvert(double ADC)
{
  double voltage;
  //cout <<ADC<<" in Decimal: " << ADC << endl;
  voltage = ADC * (5.00/65535.00);
  cout << "Voltage: " << voltage <<" V";

  //calculate current over resistor
  double current;
  current = (5.00-voltage)/1000.00;
  cout << "\t Current: " << current <<" A"<< endl;

  //calculate resistance
  double resistance;
  resistance = voltage/current;
  cout << "Resistnace: " << resistance <<" Ω"<< endl;

  return resistance;
}
```

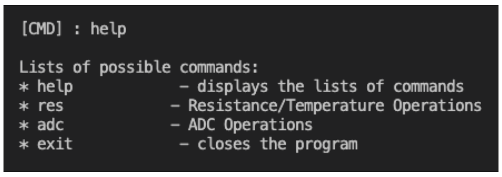
The ADCconvert() function runs the values from the variable input ADC and presents the outputs when the “adc” command is called.



```cpp
int main(){
  string cmd, tempIn;

  double rVal, tempOut;
  double T_0 = Celsius2Kelvin(20); //293.15; // 20 degree C in Kelvin
  //cout<<"Temp_0 : "<< T_0 << endl;

  do{
    cout<<"\n[CMD] : ";
    getline(cin, cmd);
```

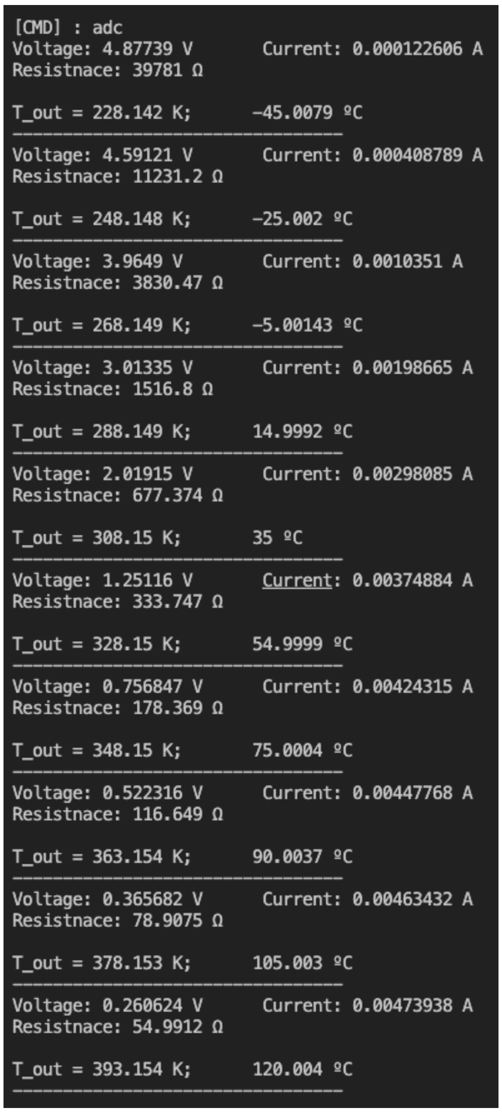
The main() function begins with an initialization of the value for T_0 followed by requesting the user the command inputs needed to run the program.

C++ Code Output Review



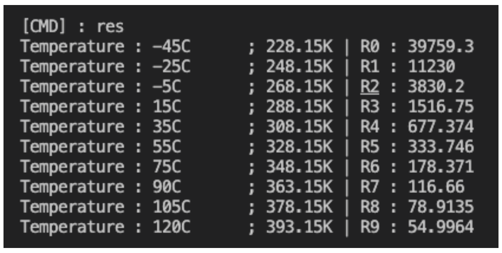
```
[CMD] : help

Lists of possible commands:
* help          - displays the lists of commands
* res          - Resistance/Temperature Operations
* adc          - ADC Operations
* exit          - closes the program
```

The program requests for a valid command in order to perform certain operations. The “help” command displays all possible commands that the program can read as valid.



```
[CMD] : adc
Voltage: 4.87739 V      Current: 0.000122606 A
Resistnace: 39781 Ω

T_out = 228.142 K;      -45.0079 ºC
----------------------------------
Voltage: 4.59121 V      Current: 0.000408789 A
Resistnace: 11231.2 Ω

T_out = 248.148 K;      -25.002 ºC
----------------------------------
Voltage: 3.9649 V       Current: 0.0010351 A
Resistnace: 3830.47 Ω

T_out = 268.149 K;      -5.00143 ºC
----------------------------------
Voltage: 3.01335 V      Current: 0.00198665 A
Resistnace: 1516.8 Ω

T_out = 288.149 K;      14.9992 ºC
----------------------------------
Voltage: 2.01915 V      Current: 0.00298085 A
Resistnace: 677.374 Ω

T_out = 308.15 K;       35 ºC
----------------------------------
Voltage: 1.25116 V      Current: 0.00374884 A
Resistnace: 333.747 Ω

T_out = 328.15 K;       54.9999 ºC
----------------------------------
Voltage: 0.756847 V     Current: 0.00424315 A
Resistnace: 178.369 Ω

T_out = 348.15 K;       75.0004 ºC
----------------------------------
Voltage: 0.522316 V     Current: 0.00447768 A
Resistnace: 116.649 Ω

T_out = 363.154 K;      90.0037 ºC
----------------------------------
Voltage: 0.365682 V     Current: 0.00463432 A
Resistnace: 78.9075 Ω

T_out = 378.153 K;      105.003 ºC
----------------------------------
Voltage: 0.260624 V     Current: 0.00473938 A
Resistnace: 54.9912 Ω

T_out = 393.154 K;      120.004 ºC
----------------------------------
```

The “adc” command allows the program to enter into adc mode where the values within the range that the temp sensor is mapped to can be plugged into an equation that can solve and output values for Voltage, Current, Resistance and the Temperature Out value.



```
[CMD] : res
Temperature : -45C      ; 228.15K | R0 : 39759.3
Temperature : -25C      ; 248.15K | R1 : 11230
Temperature : -5C       ; 268.15K | R2 : 3830.2
Temperature : 15C       ; 288.15K | R3 : 1516.75
Temperature : 35C       ; 308.15K | R4 : 677.374
Temperature : 55C       ; 328.15K | R5 : 333.746
Temperature : 75C       ; 348.15K | R6 : 178.371
Temperature : 90C       ; 363.15K | R7 : 116.66
Temperature : 105C      ; 378.15K | R8 : 78.9135
Temperature : 120C      ; 393.15K | R9 : 54.9964
```

The “res” command displays the range of temperatures used in the program and the equivalent resistance values that would represent the temp sensor resistor in the circuit.

Implementing a Spreadsheet of Values for Output Comparison

A spreadsheet was created which also simulates the input values for the temperature process and provides a detailed output of the values that the program is expected to reflect. This allows the collaborators of the project to detect any differences in the code implementation and to see if the equations follow with the concept as shown in the instructions sheet of the project. This spreadsheet is also able to conclude the differences in values by which method was able to arrive at a more consistent value.

The spreadsheet can be observed in **Appendix B** of the paper.

Documentation Link

The GitHub repository for the project is made publicly available through the following:

GitHub Link	https://github.com/jaykempis/engg156-projects-cck/tree/main/Project2_TempSensor

References

Nelson Darwin Pak (2022) *simulation of ADC in multisim | Analog to digital converter in multisim*. Video. Retrieved from https://www.youtube.com/watch?v=4l_bf2WTAGQ

SiLRing (2018) *Mock temperature sensor*. MultisimLive Circuit. Retrieved from https://www.multisim.com/content/mFPztkcsMgxCMxX7QR82xn/mock-temperature-sensor/

APPENDIX A

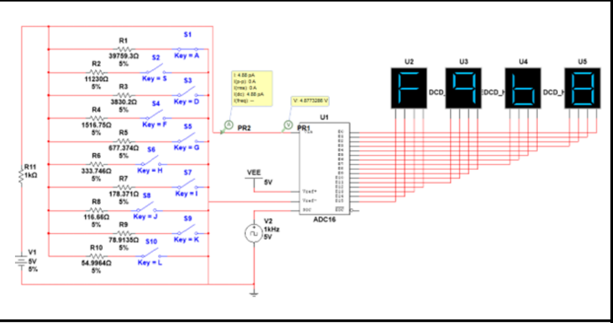
Temp Sensor Variations using a manual switch to change between different temperature values.

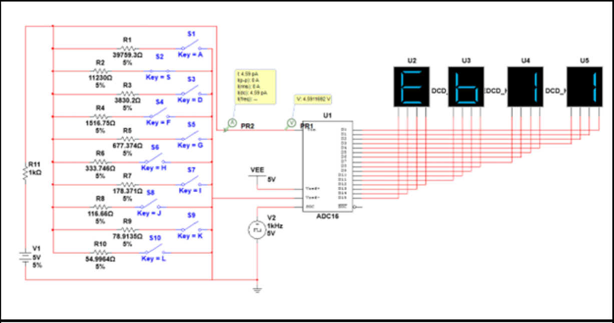
These values can be represented in Celsius:

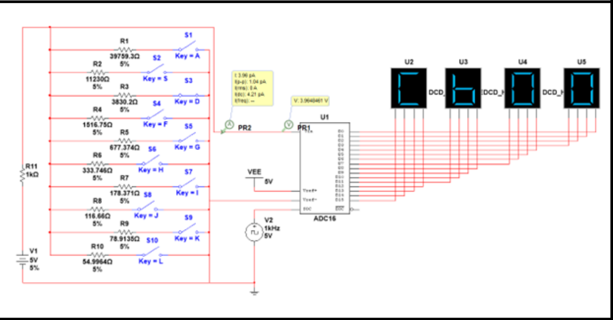
```
{-45, -25, -5, 15, 35, 55, 75, 90, 105, 120};
```

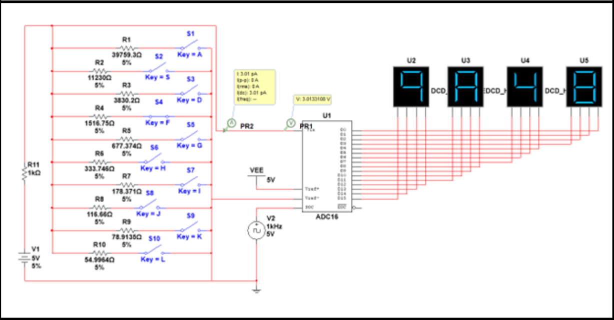
Or in their equivalent values that can be read by the circuit output

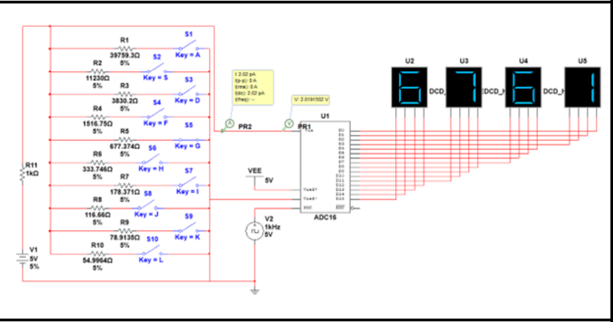
```
{"F9B8", "EB11", "CB00", "9A48", "6761","400F", "26C0", "1ABE", "12B9", "0D58"};
```

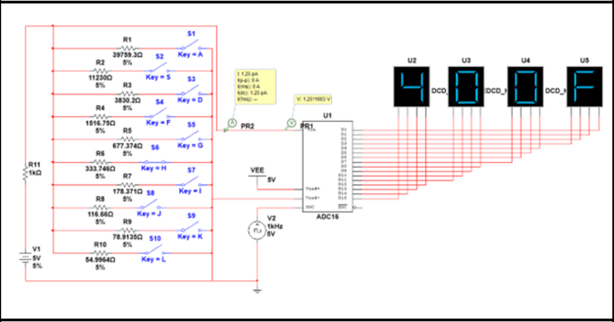


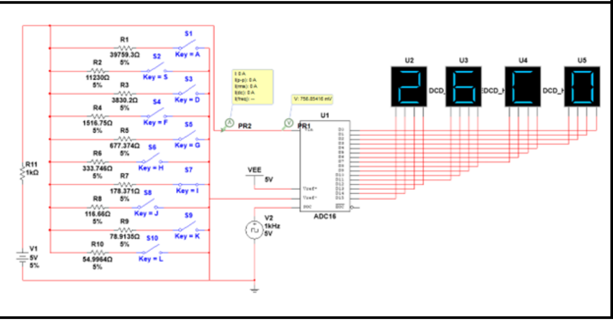


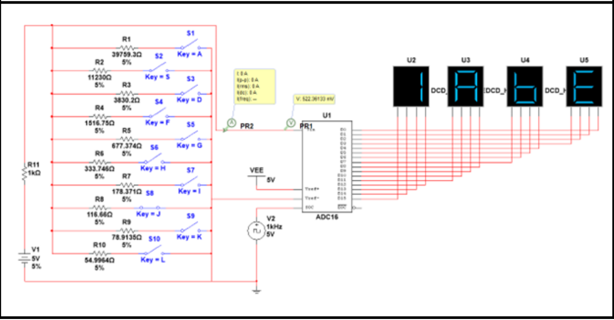


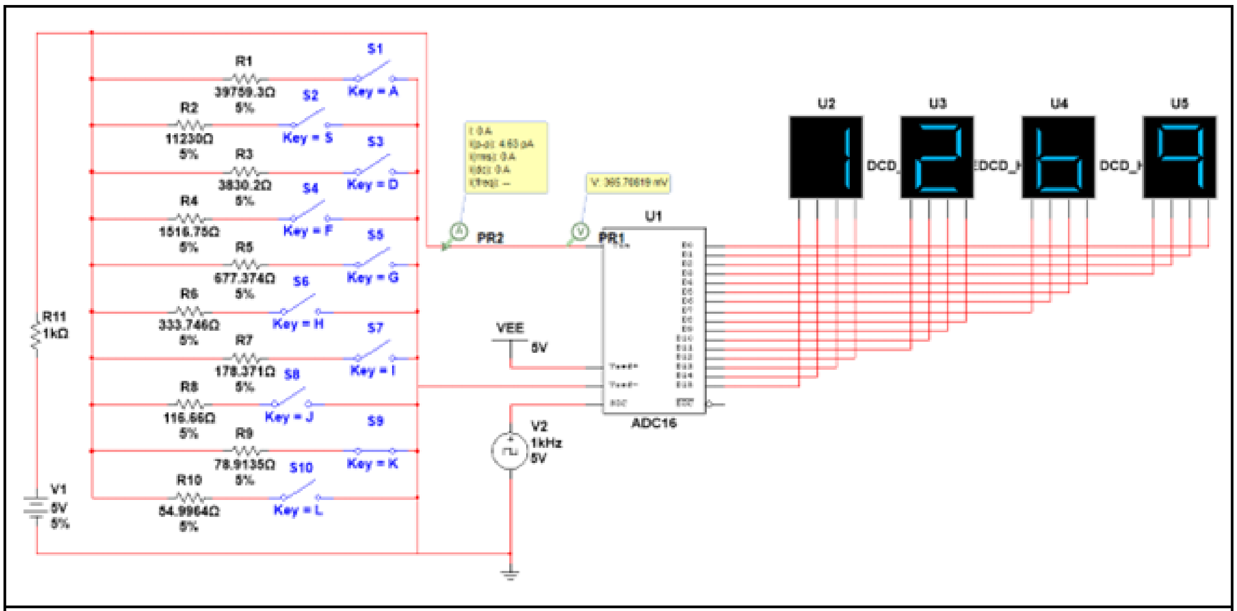


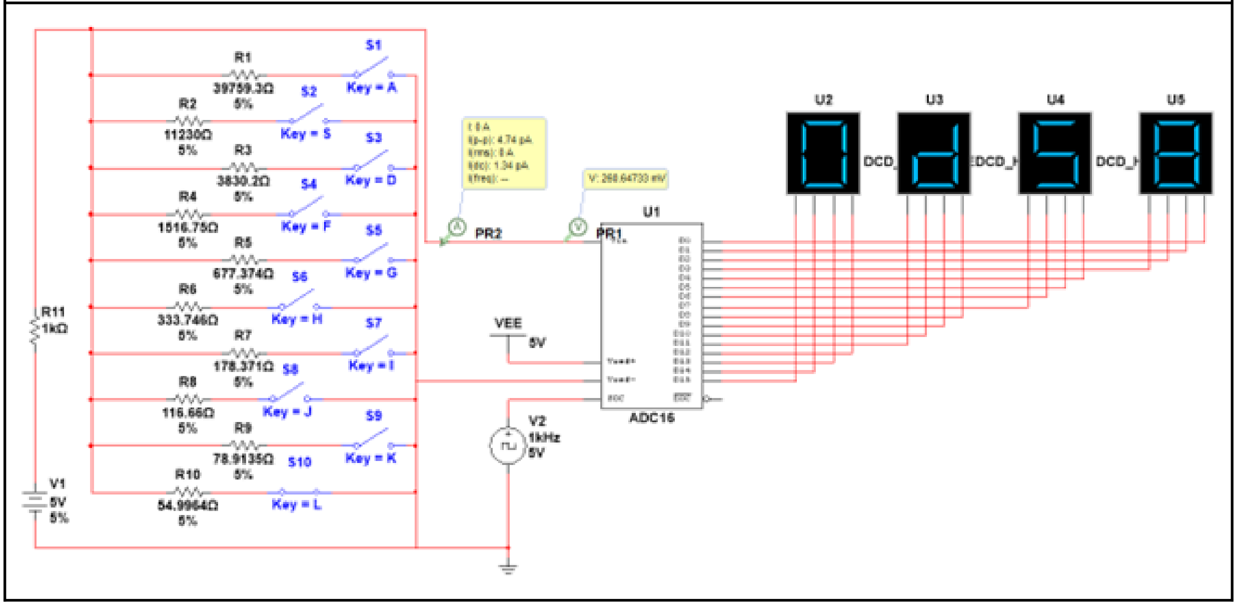












In this case, as each switch is toggled ON one at a time, the circuit receives this input as a temperature stimulus that the ADC would interpret into the hex values and output through the seven-segment displays.

APPENDIX B

Temperature Comparison Spreadsheet

T0	293.15
b	3578.825
R0	1227.196147

Temperature [C]	Temperature [K]	Resistance	Input Voltage	ADC Output	Code Resistance	Code Temp[K]	Code Temp[C]	Difference [K]	Difference [C]	Comparison of Difference
-45	228.15	39759.27266	4.8773	F9B8	39759.3	228.142	-45.0079	0.008	0.0079	Lower Diff in Celsius
-25	248.15	11229.95883	4.5912	EB11	11230	248.148	-25.002	0.002	0.002	Lower Diff in Kelvin
-5	268.15	3830.199097	3.9648	CB00	3830.2	268.149	-5.00143	0.001	0.00143	Lower Diff in Kelvin
15	288.15	1516.748766	3.0133	9A48	1516.75	288.149	14.9992	0.001	0.0008	Lower Diff in Celsius
35	308.15	677.3736071	2.0191	6761	677.374	308.15	35	0	0	Lower Diff in Celsius
55	328.15	333.7464964	1.251	400F	333.746	328.15	54.9999	0	0.0001	Lower Diff in Kelvin
75	348.15	178.3711438	0.756855	26C0	178.371	348.15	75.0004	0	0.0004	Lower Diff in Kelvin
90	363.15	116.6604009	0.5223629	1ABE	116.66	363.154	90.0037	0.004	0.0037	Lower Diff in Celsius
105	378.15	78.91354199	0.3657084	12B9	78.9135	378.153	105.003	0.003	0.003	Lower Diff in Celsius
120	393.15	54.99641164	0.2606474	0D58	54.9964	393.154	120.004	0.004	0.004	Lower Diff in Celsius