CSE 102 – Studio 3

Analog Inputs

We will be using a TMP36 temperature sensor manufactured by Analog Devices. The pinout is shown on Figure 4 of the datasheet, which is replicated below.

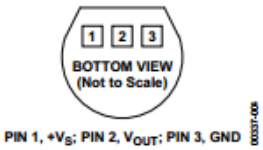

Figure 4. T-3 (TO-92)

You should connect pin 1 to +5V, pin 2 to an analog input pin, and pin 3 to GND.

Write a program called `temperature` that reads the temperature sensor and prints the value to the desktop computer. If using the `INTERNAL` analog voltage reference of 1.1V,

```
analogReference(INTERNAL);
```

a reasonably well calibrated temperature signal can be realized with the following:

```
temp = (float)(analogRead(analogInPin))*100*1.1/1024 - 50;
```

Using the PC for a display, show both unfiltered and filtered versions of the temperature. It also might be informative to show raw counts from the `analogRead()` function as well. How much filtering (i.e., how many values need to be averaged) is necessary to get a reasonably stable reading?

Change the ambient temperature of the sensor and take notice of how quickly the readings change (both the unfiltered and filtered values).

Using the graph of Figure 6 in the datasheet, can you recreate the calibration code we provided above?

Binary Output

Once you have chosen a reasonable number of values to average for a stable temperature reading, alter your temperature program to only output the filtered temperature. Assign this temperature value to an integer variable, e.g.,

```
int t = 0;       //declaration of t as integer temperature

t = (int)temp;   //assignment of float temp to int t (in loop(){})
```

Print `t` both in integer and binary format (you can utilize the second parameter of the Serial.println() function). Manually convert the binary temperature value into decimal and confirm that the two are equivalent.

What (qualitatively) will the following assignment to `t` do differently?

```
t = (int)(temp + 0.5);
```

Show your running `temperature` program to an instructor or TA.

Binary Display

Author code that counts by 1 each iteration of `loop()` between 0 and 7 (the count after 7 recycles back to 0). Spend 1 second within each iteration. (For this exercise you may use δt timing or delay(), either is fine.)

Print the value of the count on the desktop PC each iteration, both in decimal and in binary.

Designate 3 output LEDs as digits 0 (the least significant digit), 1, and 2 (the most significant digit) of the count, and output the value of the count on these LEDs. A simple way to do this is to use brute force testing of each digit. E.g.,

```
if (count & B00000001)
      digitalWrite(digitPin[0], HIGH);
else
      digitalWrite(digitPin[0], LOW);
if (count & B00000010)
      digitalWrite(digitPin[1], HIGH);
else
      digitalWrite(digitPin[1], LOW);
if (count & B00000100)
      digitalWrite(digitPin[2], HIGH);
else
      digitalWrite(digitPin[2], LOW);
```

What would be a better way to accomplish this output?

Alternate Encoding

An alternative method of encoding a value using digital information is called “one-hot” encoding. In this method, only one LED is on and the remaining LEDs are all off.

Designate each of your output LEDs as one of the values 0 through 7. Each iteration, ensure that only one of the LEDs is on and the others are all off. The LED that is on should be the one that represents the current count.