Exercise 9-12

Jonas Pedersen, Søren Graae

January 9, 2025

Exercise 9

To convert the analog input to a temperature outputted by the LM32 Temperature sensor, we can use the formula:
`float temp = ((float)A0 * (5000.0f/1024.0f))/10.0f`. To break it down: `A0` is the input from the ADC, 5000 is because VREF is $5000mV$, 1024 is the resolution, 10 comes from the resolution, which is $10mV$.

The output of the code is as below.

```
char c;
int i = 0;
c = '0' + i*2; // c = 48 + 0*2 = '48'
Serial.print(c); // Prints '48'
Serial.write(176); // Prints '?' as the terminal does
   not support the extended ASCII codes. Should be a
   degree sign.

i = 1;
c = '0' + i*2; // c = 48 + 1*2 = 50
Serial.print(c); // Prints '50'
Serial.write(176);

i = 2;
c = '0' + i*2; // c = 48 + 2*2 = 52
Serial.print(c); // Prints '52'
Serial.write(176);

i = 3;
c = '0' + i*2; // c = 48 + 3*2 = 54
Serial.print(c); // Prints '54'
Serial.write(176);
```

`Serial.write()` has no formatting; i.e. it will send the raw byte. `Serial.print()` does have formatting; i.e. `...print(61)` will send '6', '1', and '\n'.

```
/**
* @brief Connect the legs of the LM35 as described below. Connect to ATmega328P via USART.
*/

// Connect VDD to 5V
#define tempPin 0 // The output of the sensor is connected to A0
// Connect GND to GND

uint16_t sensorReading;
float temp;

void setup() {
  Serial.begin(9600);
}

void loop() {
  sensorReading = analogRead(tempPin);

  // Convert reading to degrees Celcius
  temp = ((float)sensorReading * (5000.0f / 1024.0f)) / 10.0f;

  Serial.print(temp);
  Serial.write(176);
  Serial.println("C");

  delay(1000); // So we can actually read the output
}
```

Exercise 10

```
/**
* @brief Connect the legs of the LM35 as described below. Connect to ATmega328P via USART.
* Connect three LEDs as described by their defines.
*/

```

```
6  // Connect VDD to 5V
7  #define tempPin 0 // The output of the sensor is
       connected to A0
8  // Connect GND to GND
9
10 #define greenLED 8  // The green LED is connected to D8
11 #define yellowLED 9  // The yellow LED is connected to
       D9
12 #define redLED 10  // The red LED is connected to D10
13
14 uint16_t sensorReading;
15 float temp;
16
17 void cold() {
18   digitalWrite(greenLED, HIGH);
19   digitalWrite(yellowLED, LOW);
20   digitalWrite(greenLED, LOW);
21 }
22
23 void warm() {
24   digitalWrite(greenLED, LOW);
25   digitalWrite(yellowLED, HIGH);
26   digitalWrite(redLED, LOW);
27 }
28
29 void hot() {
30   digitalWrite(greenLED, LOW);
31   digitalWrite(yellowLED, LOW);
32   digitalWrite(redLED, HIGH);
33 }
34
35 void updateLEDS() {
36   if (temp < 25)
37     cold();
38   else if (temp < 30)
39     warm();
40   else
41     hot();
42 }
43
44 void setup() {
45   pinMode(redLED, OUTPUT);
46   pinMode(greenLED, OUTPUT);
```

```
47    pinMode(yellowLED, OUTPUT);
48
49    Serial.begin(9600);
50  }
51
52  void loop() {
53    sensorReading = analogRead(tempPin);
54
55    // Convert reading to degrees Celcius
56    temp = ((float)sensorReading * (5000.0f / 1024.0f)) /
          10.0f;
57
58    Serial.print(temp);
59    Serial.write(176);
60    Serial.println("C");
61
62    updateLEDS();
63
64    delay(1000); // So we can actually read the output
65  }
```

Exercise 11

I^2C is a multi-slave *and* multi-master serial communication protocol. A master (typically a micro-controller) sends out an address on a data line, all slaves listen for their own address, the appropriate slave responds to the following command, and optionally sends data back for the master to read, dependent on the R/W bit.

To save power, refrain from updating the whole display and just update the affected area. In our case, we just update the temperature digits.

```
1  #include <Wire.h>
2  #include <LiquidCrystal_I2C.h>
3
4  // Connect VDD to 5V
5  #define tempPin 0 // The output of the sensor is
       connected to A0
6  // Connect GND to GND
7
8  LiquidCrystal_I2C lcd(0x27,16,2);  // set the LCD
       address to 0x27 for a 16 chars and 2 line display
```

```
9
10 uint16_t sensorReading;
11 float temp;
12 uint8_t warningFlag = 0;
13
14 void setup()
15 {
16   lcd.init();
17
18   lcd.backlight();
19   lcd.setCursor(0,0);
20   lcd.print("Temp in Celcius:");
21   lcd.setCursor(0,1);
22 }
23
24 void loop()
25 {
26   sensorReading = analogRead(tempPin);
27
28   // Convert reading to degrees Celcius
29   temp = ((float)sensorReading * (5000.0f / 1024.0f)) /
       10.0f;
30
31   // Print hot warning if temperature gets too high -
       only if it's not already printed
32   if (temp > 27 && warningFlag == 0) {
33     lcd.setCursor(10, 1);
34     lcd.print("!!!");
35     lcd.setCursor(0, 1);
36
37     warningFlag = 1;
38   }
39
40   // Clear hot warning if temperature drops
41   if (temp < 27 && warningFlag == 1) {
42     lcd.setCursor(10, 1);
43     lcd.print("   ");
44     lcd.setCursor(0, 1);
45
46     warningFlag = 0;
47   }
48
49   lcd.setCursor(0, 1);
```

```
50    lcd.print(temp, 2);
51
52    /**
53     * Instead of using delay(1000) for a stable
          temperature we could use a e.g. a low-pass filter
          or a moving average filter to make the temperature
           stable.
54     * If a low-pass filter is to be used, a delay between
           each display update should still be implemented
          however seen fit.
55     * For a moving average filter, the display would just
           update at the end of data collection.
56     */
57    delay(1000);
58  }
```