

**Introduction to Microcontrollers**

# **Experiment 1:**

**(KEY FOR ECE STUDENTS)**

Learning the Arduino Microcontroller

# Contents

<b><u>1 Objectives</u></b> .....	<b>3</b>
<b><u>2 Background</u></b> .....	<b>3</b>
<u>2.1 Arduino Microcontroller</u> .....	3
<u>2.2 Arduino IDE</u> .....	3
2.2.1 Sketchbook .....	3
2.2.2 Uploading .....	3
2.2.3 Libraries .....	4
2.2.4 Serial Monitor .....	4
2.2.5 Toolbar Buttons .....	4
<u>2.3 Analog Temperature Sensor</u> .....	4
2.3.1 What is the TMP36? .....	4
2.3.2 Sensor Connections .....	5
2.3.2 Reading from the Sensor .....	5
<u>2.4 Digital Temperature Sensor</u> .....	5
2.4.1 What is the MCP9808? .....	5
2.4.2 I2C Communication Protocol .....	5
<b><u>3 Procedures</u></b> .....	<b>6</b>
<u>3.1 Reading from the TMP36 (Analog Sensor)</u> .....	6
3.1.1 Connecting to the Sensor .....	6
3.1.2 Reading the Analog Temperature Data .....	6
3.1.3 Programming a Simple Analog Thermometer .....	6
3.1.4 Program the Arduino .....	7
3.1.4 Collecting the Output .....	8
<u>3.2 Reading from the MCP9808 (Digital Sensor)</u> .....	8
3.2.1 Connecting to the Sensor .....	8
3.2.2 Adding the Arduino Library .....	9
3.2.3 Programming a Simple Digital Thermometer .....	9
3.2.4 Program the Arduino .....	11
3.2.5 Collecting the Output .....	11
<b><u>4 Study Questions</u></b> .....	<b>13</b>
<b><u>5 Equipment</u></b> .....	<b>13</b>

# 1 Objectives

To learn to utilize the Arduino Mega microcontroller for locally reading from analog and digital sensors.

## 2 Background

### 2.1 Arduino Microcontroller

Arduino boards are able to read inputs - light on a sensor, a finger on a button, or a Twitter message - and turn it into an output - activating a motor, turning on an LED, publishing something online. You can tell your board what to do by sending a set of instructions to the microcontroller on the board. To do so you use the Arduino programming language (based on Wiring), and the Arduino Software (IDE), based on Processing.

### 2.2 Arduino IDE

Programs written using Arduino Software (IDE) are called sketches. These sketches are written in the text editor and are saved with the file extension .ino. The editor has features for cutting/pasting and for searching/replacing text. The message area gives feedback while saving and exporting and also displays errors. The console displays text output by the Arduino Software (IDE), including complete error messages and other information. The bottom righthand corner of the window displays the configured board and serial port. The toolbar buttons allow you to verify and upload programs, create, open, and save sketches, and open the serial monitor.

#### 2.2.1 Sketchbook

The Arduino Software (IDE) uses the concept of a sketchbook: a standard place to store your programs (or sketches). The sketches in your sketchbook can be opened from the File > Sketchbook menu or from the Open button on the toolbar. The first time you run the Arduino software, it will automatically create a directory for your sketchbook. You can view or change the location of the sketchbook location from with the Preferences dialog.

#### 2.2.2 Uploading

Before uploading your sketch, you need to select the correct items from the Tools > Board and Tools > Port menus. The boards are described below. On Windows, it's probably COM1 or COM2 (for a serial board) or COM4, COM5, COM7, or higher (for a USB board) - to find out, you look for USB serial device in the ports section of the Windows Device Manager. Once you've selected the correct serial port and board, press the upload button in the toolbar or select the Upload item from the Sketch menu. Current Arduino boards will reset automatically and begin the upload. On most boards, you'll see the RX and TX LEDs blink as the sketch is uploaded. The Arduino Software (IDE) will display a message when the upload is complete, or show an error.

When you upload a sketch, you're using the Arduino bootloader, a small program that has been loaded on to the microcontroller on your board. It allows you to upload code without using any additional hardware. The bootloader is active for a few seconds when the board resets; then it starts whichever sketch was most recently uploaded to the microcontroller. The bootloader will blink the on-board (pin 13) LED when it starts (i.e. when the board resets).







### 2.2.3 Libraries

Libraries provide extra functionality for use in sketches, e.g. working with hardware or manipulating data. To use a library in a sketch, select it from the Sketch > Import Library menu. This will insert one or more `#include` statements at the top of the sketch and compile the library with your sketch. Because libraries are uploaded to the board with your sketch, they increase the amount of space it takes up. If a sketch no longer needs a library, simply delete its `#include` statements from the top of your code.

### 2.2.4 Serial Monitor

This displays serial sent from the Arduino or Genuino board over USB or serial connector. To send data to the board, enter text and click on the "send" button or press enter. Choose the baud rate from the drop-down menu that matches the rate passed to `Serial.begin` in your sketch. Note that the board will reset when you connect with the serial monitor.

### 2.2.5 Toolbar Buttons

Button	Button Functionality
	Checks your code for errors compiling it.
	Compiles your code and uploads it to the configured board.
	Creates a new sketch.
	Presents a menu of all the sketches in your sketchbook.
	Saves your sketch.
	Opens the serial monitor.

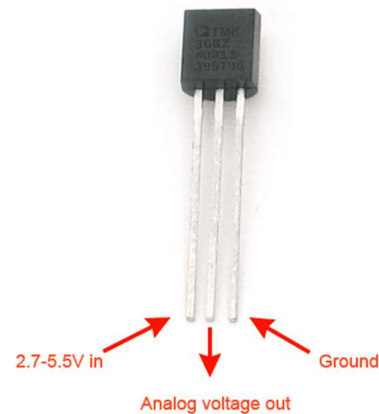
## 2.3 Analog Temperature Sensor

### 2.3.1 What is the TMP36?

These sensors use a solid-state technique to determine the temperature. That is to say, they don't use mercury (like old thermometers), bimetallic strips (like in some home thermometers or stoves), nor do they use thermistors (temperature sensitive resistors). Instead, they use the fact as temperature increases, the voltage across a diode increases at a known rate. (Technically, this is actually the voltage drop between the base and emitter - the  $V_{be}$  - of a transistor.) By precisely amplifying the voltage change, it is easy to generate an analog signal that is directly proportional to temperature.

### 2.3.2 Sensor Connections

Using the TMP36 is easy, simply connect the left pin to power (2.7-5.5V) and the right pin to ground. Then the middle pin will have an analog voltage that is directly proportional (linear) to the temperature. The analog voltage is independent of the power supply.



### 2.3.2 Reading from the Sensor

To convert the voltage to temperature, simply use the basic formula:

$$\text{Temp in } ^\circ\text{C} = [(V_{\text{out in mV}}) - 500] / 10$$

So for example, if the voltage out is 1V that means that the temperature is  $((1000 \text{ mV} - 500) / 10) = 50 ^\circ\text{C}$

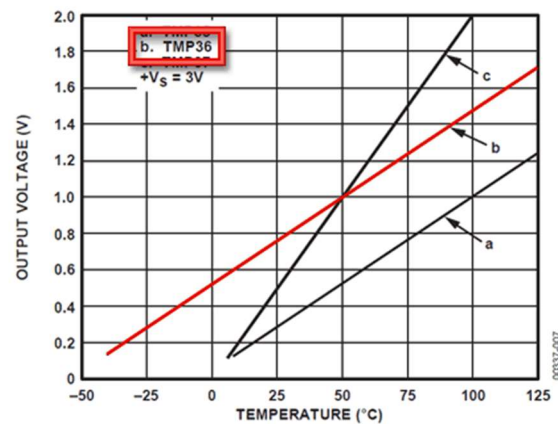


Figure 6. Output Voltage vs. Temperature

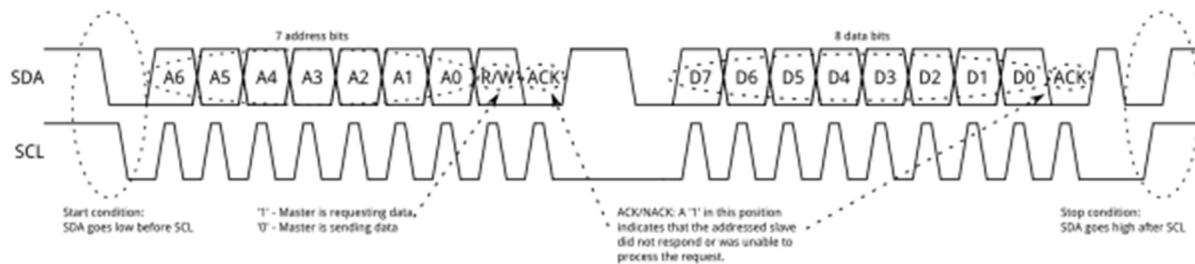
## 2.4 Digital Temperature Sensor

### 2.4.1 What is the MCP9808?

The MCP9808 digital temperature sensor relies on I2C for communication and has a typical accuracy of  $\pm 0.25^\circ\text{C}$  over the sensor's  $-40^\circ\text{C}$  to  $+125^\circ\text{C}$  range and precision of  $+0.0625^\circ\text{C}$ . They work great with any microcontroller using standard i2c. There are 3 address pins so you can connect up to 8 to a single I2C bus without address collisions.

### 2.4.2 I2C Communication Protocol

Communication via I2C is more complex than with a UART or SPI solution. The signalling must adhere to a certain protocol for the devices on the bus to recognize it as valid I2C communications.



Messages are broken up into two types of frame: an address frame, where the master indicates the slave to which the message is being sent, and one or more data frames, which are 8-bit data messages passed from master to slave or vice versa. Data is placed on the SDA line after SCL goes low, and is sampled after the SCL line goes high. The time between clock edge and data read/write is defined by the devices on the bus and will vary from chip to chip.

## 3 Procedures

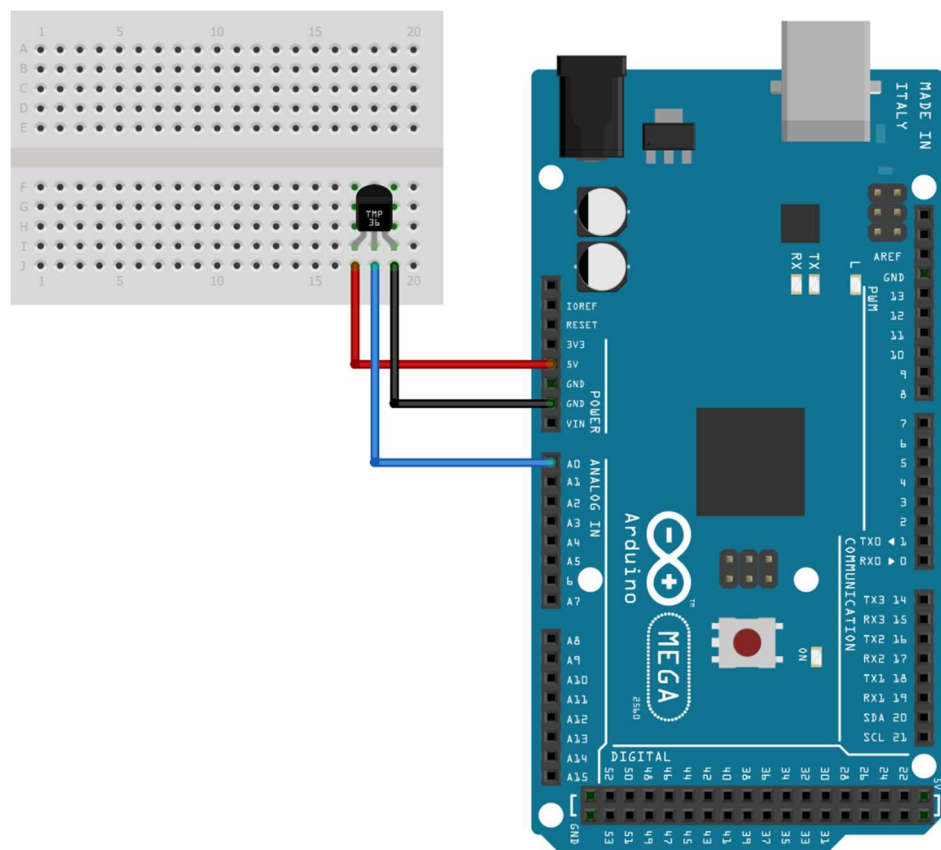
### 3.1 Reading from the TMP36 (Analog Sensor)

#### 3.1.1 Connecting to the Sensor

These sensors have little chips in them and while they're not that delicate, they do need to be handled properly. Be careful of static electricity when handling them and make sure the power supply is connected up correctly and is between 2.7 and 5.5V DC.

They come in a "TO-92" package which means the chip is housed in a plastic hemi-cylinder with three legs. The legs can be bent easily to allow the sensor to be plugged into a breadboard.

To read from the TMP36, connect the left pin to the 5V female pin on the Arduino Mega, connect the middle pin to A0 on the Arduino Mega, and finally connect the right pin to the GND pin on the Arduino Mega.



### 3.1.2 Reading the Analog Temperature Data

Since we're using a 5V Arduino Mega, and connecting the sensor directly into an Analog pin, you can use these formulas to turn the 10-bit analog reading into a temperature:

$$\text{Voltage at pin in milliVolts} = (\text{reading from ADC}) * (5000/1024)$$

This formula converts the number 0-1023 from the ADC into 0-5000mV (= 5V)

Then, to convert millivolts into temperature, use this formula:

$$\text{Centigrade temperature} = [(\text{analog voltage in mV}) - 500] / 10$$

### 3.1.3 Programming a Simple Analog Thermometer

This example code for Arduino shows a quick way to create an analog temperature sensor, it simply prints to the serial port what the current temperature is in both Celsius and Fahrenheit.

```
//TMP36 Pin Variables
int sensorPin = 0; //the analog pin the TMP36's Vout (sense) pin is connected
to
//the resolution is 10 mV / degree centigrade with a
//500 mV offset to allow for negative temperatures

/*
 * setup() - this function runs once when you turn your Arduino on
 * We initialize the serial connection with the computer
 */
void setup()
{
    Serial.begin(9600); //Start the serial connection with the computer
                        //to view the result open the serial monitor
}

void loop()           // run over and over again
{
    //getting the voltage reading from the temperature sensor
    int reading = analogRead(sensorPin);

    // converting that reading to voltage, for 3.3v arduino use 3.3
    float voltage = reading * 5.0;
    voltage /= 1024.0;

    // print out the voltage
    Serial.print(voltage); Serial.println(" volts");

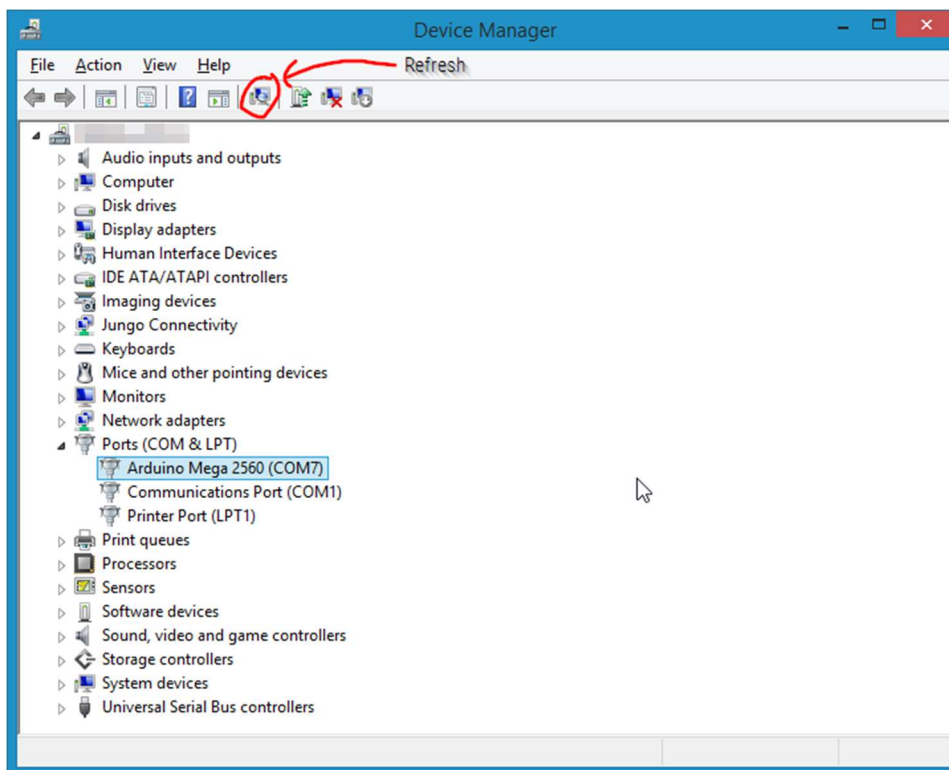
    // now print out the temperature
    float temperatureC = (voltage - 0.5) * 100 ; //converting from 10 mv per
    degree wit 500 mV offset
                                                //to degrees ((voltage -
    500mV) times 100)
    Serial.print(temperatureC); Serial.println(" degrees C");
}
```

```
// now convert to Fahrenheit
float temperatureF = (temperatureC * 9.0 / 5.0) + 32.0;
Serial.print(temperatureF); Serial.println(" degrees F");

delay(1000); //waiting a second
}
```

### 3.1.4 Program the Arduino

Enter the code into a new sketch within the Arduino IDE. Verify that under tools the board type is set to “Arduino/Genuino Mega or Mega 2560” and the port is set to the COM device that appears in the device manager after the Arduino Mega is connected.



If the device does not show under Ports (COM & LPT) then refresh the hardware list (Scan for Hardware Changes).

Once the device type and ports selected then upload the program to the Arduino.

### 3.1.4 Collecting the Output

With the code now loaded onto the Arduino, open the Serial Monitor in the Arduino. Verify that the temperatures that are printed to the serial are correct, collect 30 samples of temperature data, and save the data to a file to append to your lab report.

## 3.2 Reading from the MCP9808 (Digital Sensor)

### 3.2.1 Connecting to the Sensor

Connect Vdd to the 5V source on the Arduino Mega.

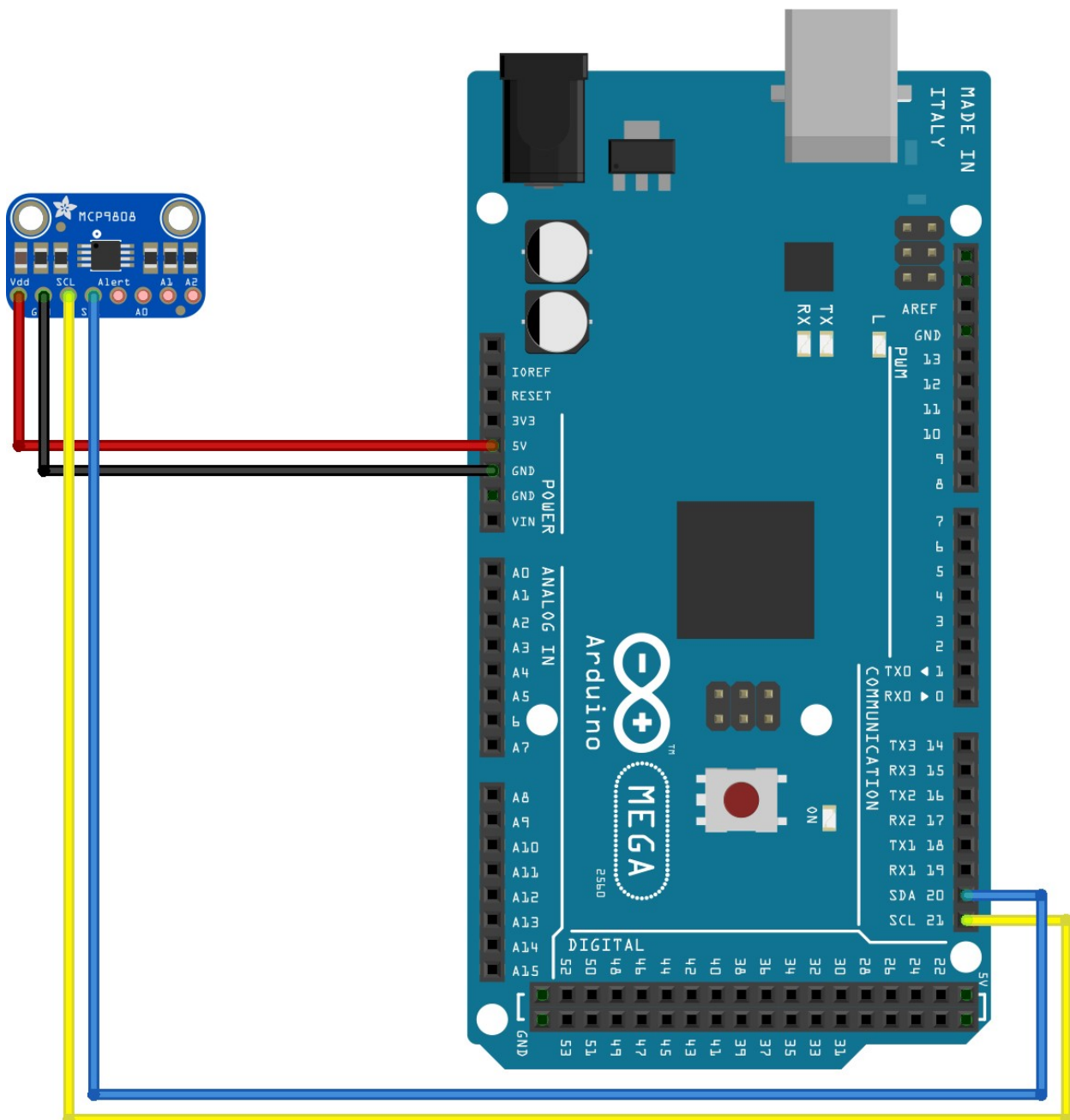


Connect GND to common power/data ground

Connect the SCL pin to the I2C clock SCL pin on your Arduino. On the Arduino Mega it is also known as digital 21.

Connect the SDA pin to the I2C data SDA pin on your Arduino. On the Arduino Mega it is also known as digital 20.

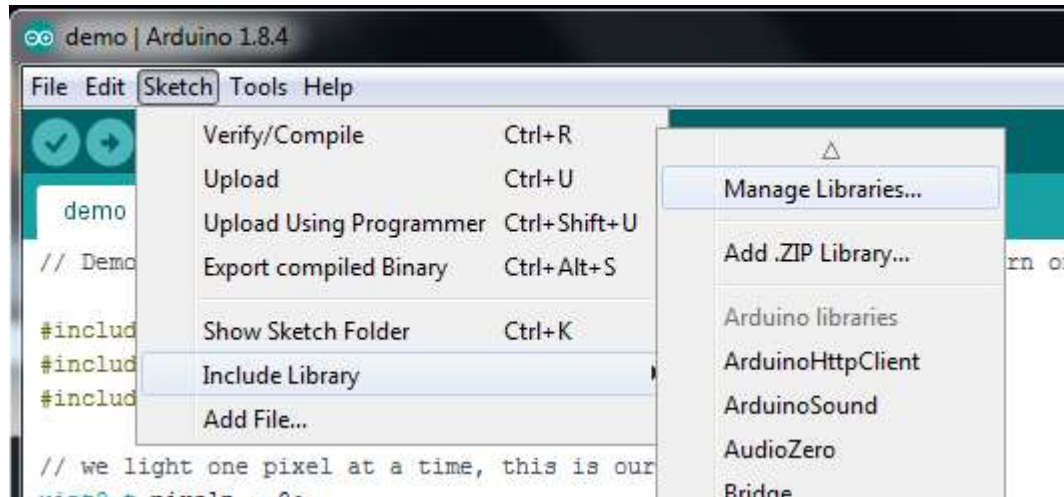
The MCP9808 has a default I2C address of 0x18 but you can set the address to any of 8 values between 0x18 and 0x1F so you can have up to 8 of these sensors all sharing the same SCL/SDA pins.



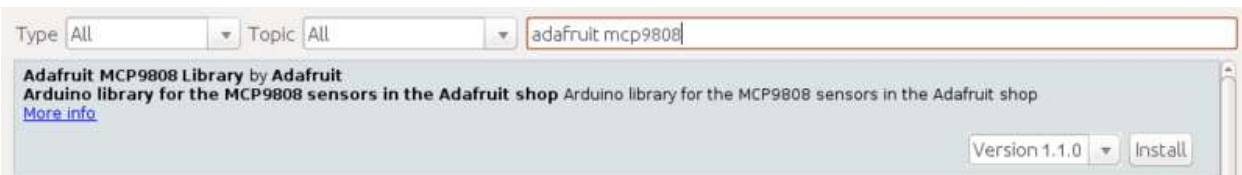
### 3.2.2 Adding the Arduino Library

To begin reading sensor data, you will need to download the Adafruit MCP9808 library from the Arduino library manager.

Open up the Arduino library manager:



Search for the Adafruit MCP9808 library and install it:



### 3.2.3 Programming a Simple Digital Thermometer

This example code for Arduino shows a quick way to create a digital temperature sensor, it simply prints to the serial port what the current temperature is in both Celsius and Fahrenheit.

```
/*!  
This is a demo for the Adafruit MCP9808 breakout  
----> http://www.adafruit.com/products/1782  
Adafruit invests time and resources providing this open source code,  
please support Adafruit and open-source hardware by purchasing  
products from Adafruit!  
*/  
  
#include <Wire.h>  
#include "Adafruit_MCP9808.h"  
  
// Create the MCP9808 temperature sensor object  
Adafruit_MCP9808 tempsensor = Adafruit_MCP9808();  
  
void setup() {  
  Serial.begin(9600);
```

```

while (!Serial); //waits for serial terminal to be open, necessary in newer
arduino boards.
Serial.println("MCP9808 demo");

// Make sure the sensor is found, you can also pass in a different i2c
// address with tempsensor.begin(0x19) for example, also can be left in
blank for default address use
// Also there is a table with all addres possible for this sensor, you can
connect multiple sensors
// to the same i2c bus, just configure each sensor with a different address
and define multiple objects for that
//  A2 A1 A0 address
//  0  0  0   0x18  this is the default address
//  0  0  1   0x19
//  0  1  0   0x1A
//  0  1  1   0x1B
//  1  0  0   0x1C
//  1  0  1   0x1D
//  1  1  0   0x1E
//  1  1  1   0x1F
if (!tempsensor.begin(0x18)) {
    Serial.println("Couldn't find MCP9808! Check your connections and verify
the address is correct.");
    while (1);
}

Serial.println("Found MCP9808!");

tempsensor.setResolution(3); // sets the resolution mode of reading, the
modes are defined in the table bellow:
// Mode Resolution SampleTime
//  0      0.5°C       30 ms
//  1      0.25°C      65 ms
//  2      0.125°C     130 ms
//  3      0.0625°C    250 ms
}

void loop() {
    Serial.println("wake up MCP9808.... "); // wake up MCP9808 - power
consumption ~200 micro Ampere
    tempsensor.wake(); // wake up, ready to read!

    // Read and print out the temperature, also shows the resolution mode used
for reading.
    Serial.print("Resolution in mode: ");
    Serial.println(tempsensor.getResolution());
    float c = tempsensor.readTempC();
    float f = tempsensor.readTempF();
    Serial.print("Temp: ");
    Serial.print(c, 4); Serial.print("*C\t and ");
    Serial.print(f, 4); Serial.println("*F.");

    delay(2000);
}

```

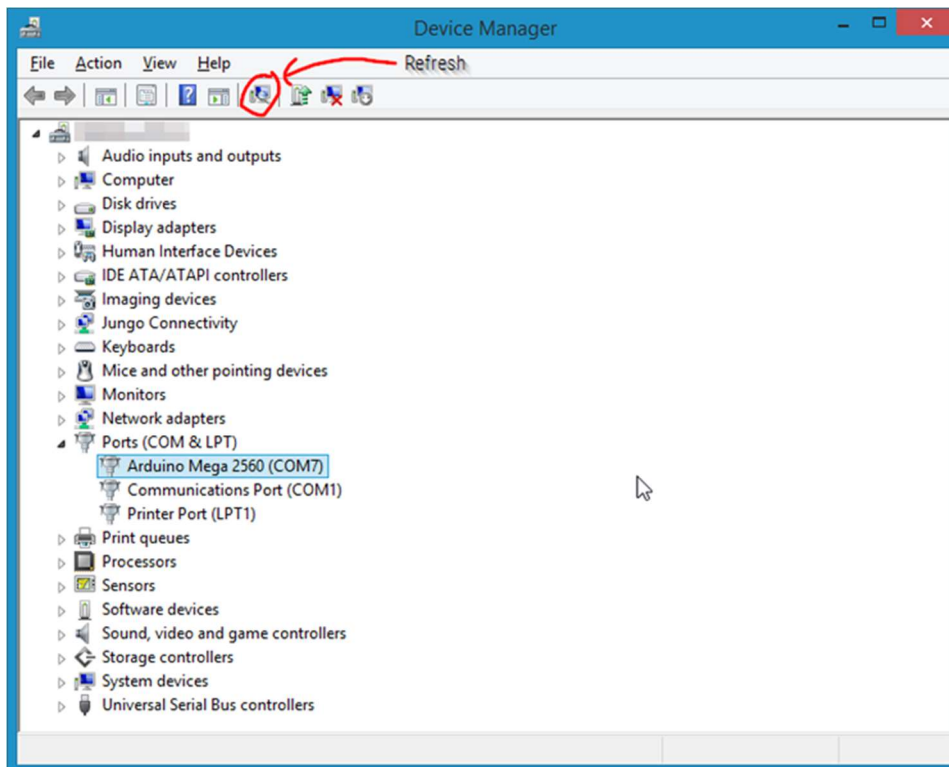
```

Serial.println("Shutdown MCP9808.... ");
tempensor.shutdown_wake(1); // shutdown MSP9808 - power consumption ~0.1
mikro Ampere, stops temperature sampling
Serial.println("");
delay(200);
}

```

### 3.2.4 Program the Arduino

Enter the code into a new sketch within the Arduino IDE. Verify that under tools the board type is set to “Arduino/Genuino Mega or Mega 2560” and the port is set to the COM device that appears in the device manager after the Arduino Mega is connected.



If the device does not show under Ports (COM & LPT) then refresh the hardware list (Scan for Hardware Changes).

Once the device type and ports selected then upload the program to the Arduino.

### 3.2.5 Collecting the Output

With the code now loaded onto the Arduino, open the Serial Monitor in the Arduino. Verify that the temperatures that are printed to the serial are correct, collect 30 samples of temperature data, and save the data to a file to append to your lab report.

## 4 Study Questions & Deliverables

1. Provide a comprehensive report that demonstrates your completion of this laboratory assignment. Key items to include in your report are introduction and background, methodologies, results and conclusions, figures, and tables.
2. Plot the temperature data recorded from the TMP36 analog temperature sensor.
3. Plot the temperature data recorded from the MCP9808 digital temperature sensor.

**Instructor:** Dr. Shayan (Sean) Taheri.

**Note – Cheating and Plagiarism:** Cheating and plagiarism are not permitted in any form and cause certain penalties. The instructor reserves the right to fail culprits.

**Deliverable:** All your responses to the assignment questions should be included in a single compressed file to be uploaded in the Gannon University (GU) – Blackboard Learn environment.

## 5 Equipment

Name	Quantity
Arduino Mega Microcontroller	1
USB-A to USB-B Cable	1
MCP9808 Temperature Sensor	1
TMP36 Analog Temperature Sensor	1
Male-to-Female Dupont Jumpers	12
Breadboard	1