Section 1: General Lab Info */0.5 points*

Microprocessors II and Embedded Systems

EECE.4800

Lab 3: Controlling an I2C Device

Yan Luo

Group 1

Grayson Colwell

Handed in: November 20, 2017

Lab Due: November 20, 2017

Section 2: Contributions /1 points

1. Group Member 1 – Grayson Colwell

Responsible for developing Main.c. Wrote the function responsible for naming the filenames of the pictures being taken by the webcam. Wrote the algorithm responsible for removing illegal characters from the time function used to name the pictures taken by the camera. Developed the loop responsible for allowing the user to select whether a picture will be taken, obtain a temperature reading from the temperature sensor and based off the temperature value, take a picture, and also allow the user to exit the program.

2. Group Member 2 – Andrew Macgregor

Responsible for developing camera.cpp and camera.h. Initialized the webcam on the video0 interface. Had the camera take a picture as the webcam is always filming and error checked the camera so that if the camera wasn't connected, the function would false and would also return false if the camera grabbed a blank image. Saved the picture taken by the webcam as a jpeg by adding the two strings together (string passed in from the main function and the string ".jpeg"). wrote the function to save the file and also to error check if there was an error in saving the picture.

3. Group Member 3 – Jose Velis

Responsible for developing temp_sensor.h. Initialized the temperature sensor registers and the I2C communication lines with the temperature sensor. Created a register to store the values returned from the temperature sensor to then send to the main program to compare with the threshold temperature. If the temperature returned by this function is greater than threshold temperature, then a picture is taken by the webcam and saved to the Galileo.

Section 3: Purpose /0.5 points

The purpose of this lab was to gain an understanding of how to utilize I2C bus protocol to control devices connected to the Galileo board. The devices under investigation in this lab are the TMP102 temperature sensor and a USB webcam used to capture images. By utilizing OpenCV, the USB webcam was very easily implemented with the Galileo.

Section 4: Introduction /0.5 points

To be successful in this lab, there were a few key concepts that needed to be understood. This lab was centered on solely using the Galileo board to implement an I2C communication line between two key devices, the USB webcam and the TMP102 temperature sensor. In order to understand how to get temperature readings from the sensor, the I2C communication lines needed to be defined so that the temperature sensor knew where to send the temperature information to. In order to get the webcam to work correctly, OpenCV needed to be utilized. Using functions from the OpenCV library, the camera was able to be used to take and save pictures to the Galileo.

Section 5: Materials, Devices and Instruments /0.5 points

- Intel Galileo Gen2 embedded computer: The embedded computer used to gain an understanding of I2C communication between devices.
- TMP102 Temperature Sensor: Temperature sensor used to send temperature readings to the Galileo embedded computer.
- USB Webcam: Webcam used to take pictures based off the value read in by the Temperature sensor.
- UART Cable: Cable used to connect to the Galileo embedded computer to find the IP Address of the device so we could connect to the board over Wifi using WinSCP.

Section 6: Schematics /0.5 points

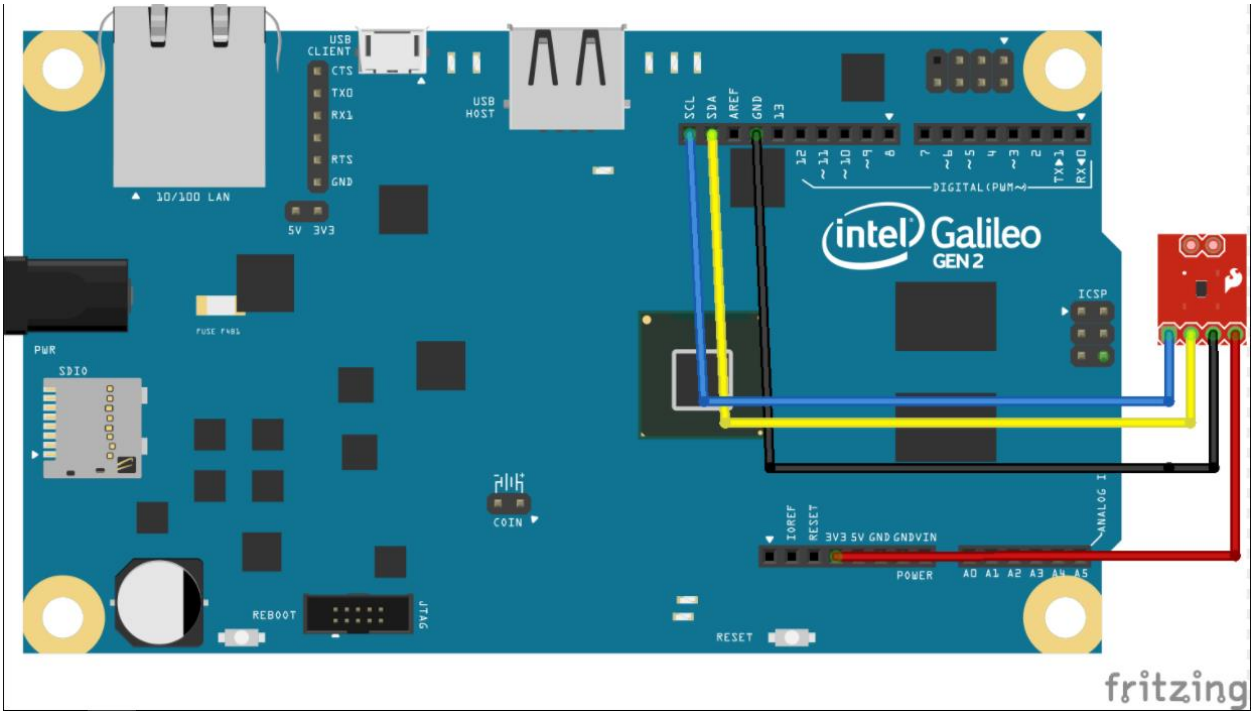

Figure 1: Schematic of how the Temperature sensor was connected to the Galileo Board.

In Figure 1, the pinout of how the Temperature sensor was connected to the Galileo Board is shown. There was hardly any hardware implementation in this lab as it was mostly focused on understanding and implementing I2C communication between the devices.

Section 7: Lab Methods and Procedure /2 points

Hardware Design:

When looking at the hardware in this lab, there was very little hardware implementation. The temperature sensor was connected to the Galileo board as described in the Instructions document.

Software Design:

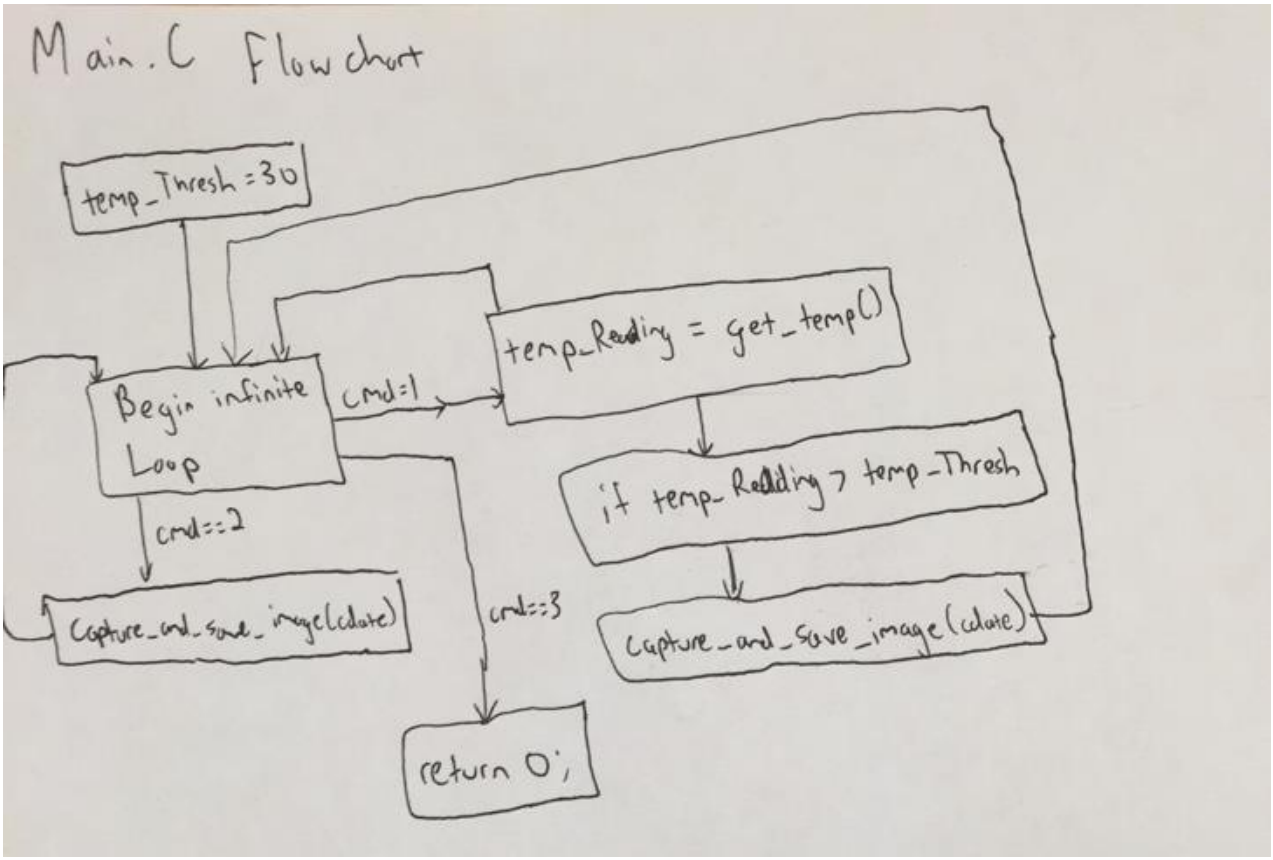

Figure 2: Flowchart describing the flow of the main program.

Illustrated in Figure 2 is the flowchart for the main module of the program implemented in this lab. The purpose of this module is to get a temperature reading from the temperature sensor and based off this value, take a picture. This module also allowed the user to take a picture using the '2' command to test and make sure that the camera module was working. In Figure 3, the code used for naming the file of the picture is shown and also the code for taking a picture based off of the temperature reading is also shown.

```
while (1)
{
        printf("\nEnter the Command you would like to do (1 temp sensor-triggered pic, 2 take a pic, 3 exit)\n");
        scanf("%d", &cmd);
        if (cmd == 1)
        {
                temp_Reading = get_temp();
                printf("Temperature:%lf degrees C\n", temp_Reading);
                date = time(NULL);
                cdate = asctime(localtime(&date));
                if (temp_Reading > temp_Thresh)
                {
                        for (int i = 0; i < 25; i++)
                        {
                                if (cdate[i] == 32)
                                {
                                        cdate[i] = '_';
                                }
                                else if (cdate[i] == 58)
                                {
                                        cdate[i] = '_';
                                }
                                else if (cdate[i] == 10)
                                {
                                        cdate[i] = '_';
                                }
                        }
printf("Taking picture: %s \n", cdate);
                        capture_and_save_image(cdate);
                } else {
printf("Not hot enough for picture -- \n\tneeds to be greater than %f\n", temp_Thresh);
```

Figure 3: Code used to name the picture taken by the Webcam and whether or not to take a picture based off the temperature reading.

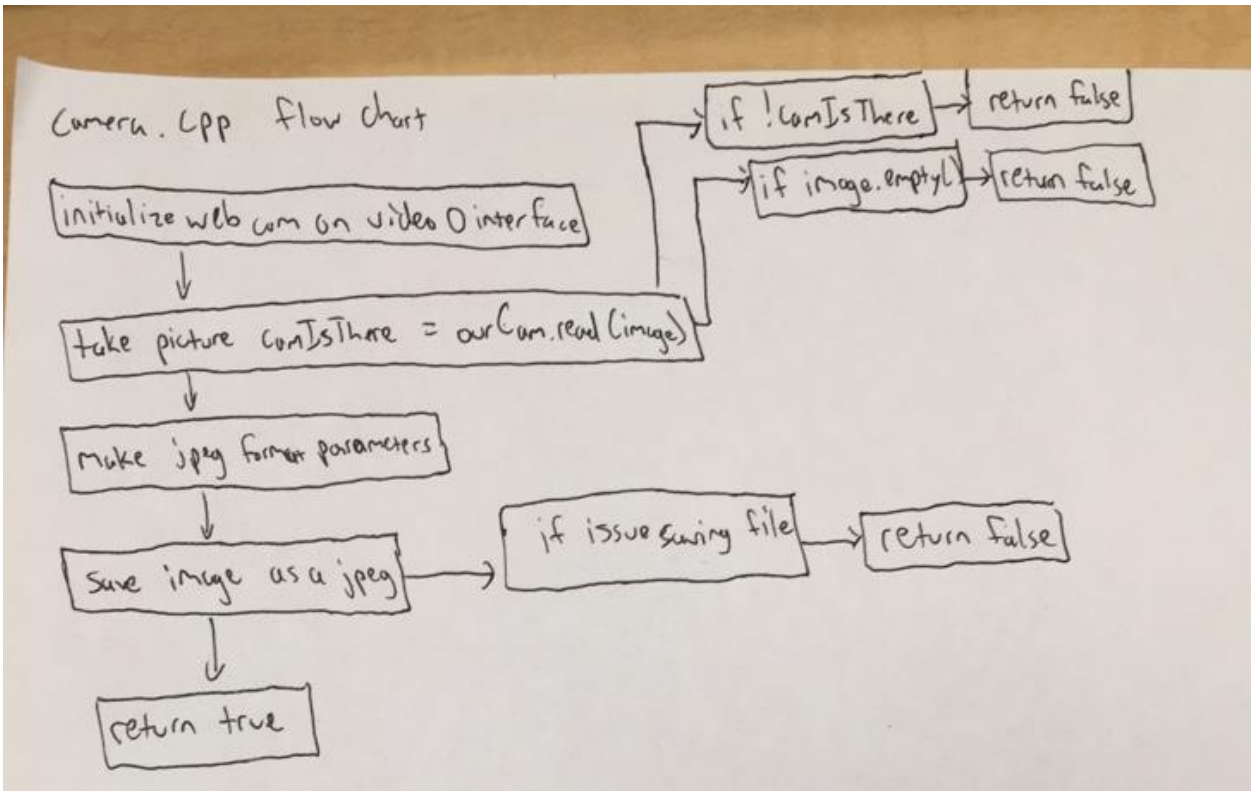

Figure 4: Flowchart of the Camera.cpp Module.

Shown in Figure 4 is the logical sequence of how the Camera module was implemented in order to successfully take a picture. The webcam is initialized on the video0 interface so that the camera is constantly taking a video. The camera then reads a single frame from video and this is the image that will be saved. The code then ensures that the camera is connected and that the image read is an empty frame. From here, the image is saved as a JPEG and if there is no error in saving the picture, the module returns true. This code is shown in Figure 5.

```
bool capture_and_save_image(char* filename)
{
  //init webcam on video0 interface
  VideoCapture ourCam;
  ourCam.open(0);

  Mat image;
  bool camIsThere = false;

  //read a frame from the vid camera into image
  camIsThere = ourCam.read(image);

  //test if camera not connected
  if(!camIsThere)
    return false;
  //test if a blank image was grabbed
  if(image.empty())
    return false;

  //make jpeg format parameters
  vector<int> compression_params;
  compression_params.push_back(IMWRITE_JPEG_QUALITY);
  compression_params.push_back(95);

  //make filename
  string fn = string(filename) + string(".jpeg");

  //Code from inWrite() example in OpenCV docs
  //try to save to file
  try{
    imwrite(fn, image, compression_params);
  }
  catch (Exception& ex) {
    cout << "Issue saving file: " << ex.what() << endl;
    return false;
  }

  return true;
```

Figure 5: Camera.cpp Module used to take a picture from the Webcam.

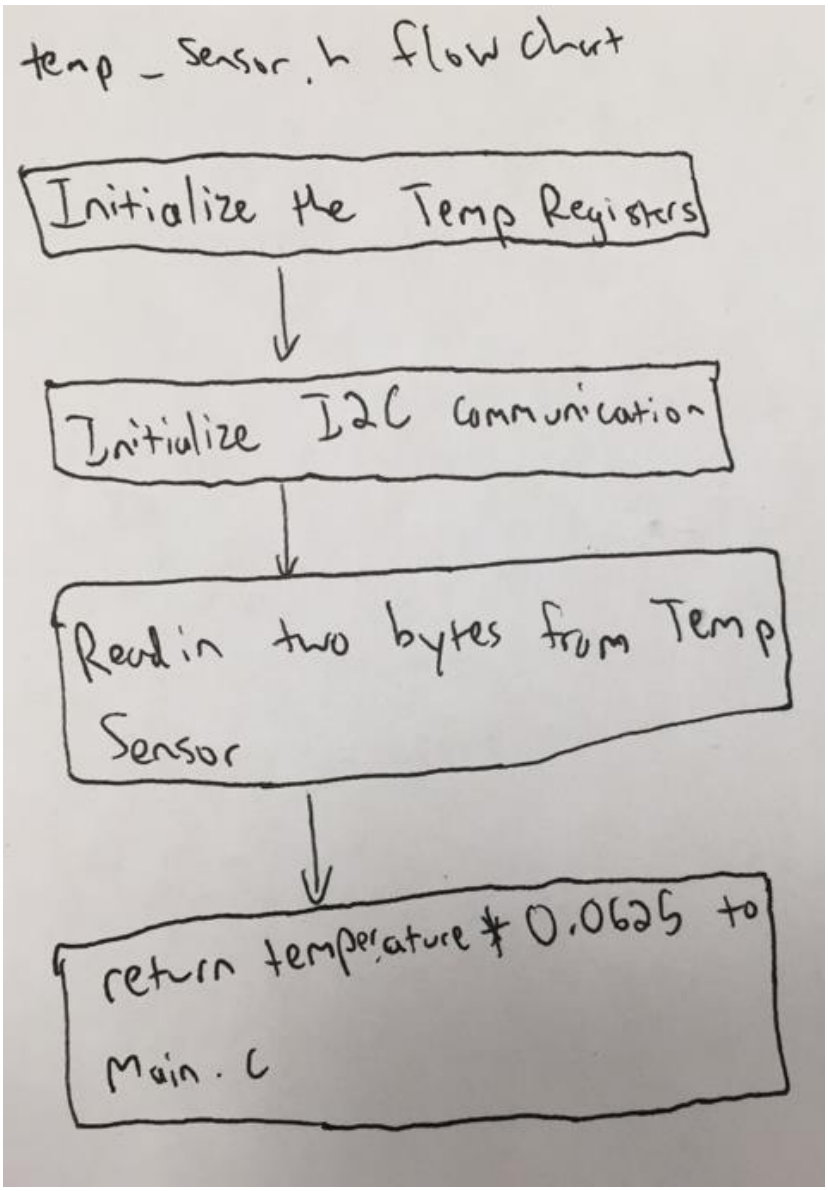

Figure 6: Flowchart of the Temperature module.

Shown in Figure 6 is the Flowchart of how the temperature sensor module was implemented. The registers of the temperature sensor were initialized and then the I2C communication was initialized between the Galileo and the Temperature sensor. After completing this, the temperature was read from the temperature sensor and sent to the Galileo board where it was compared to the threshold temperature (30 degrees Celsius). If the temperature read in was greater than this value, then the webcam would save a picture to the Galileo. Shown in Figure 7 is the actual code of how the Temperature sensor was implemented to the Galileo.

```
double get_temp(){
        I2c i2c(0);
        i2c.address(TMP102Address);

uint8_t dataReg [2];

int buffer = i2c.read(dataReg,2); // read two bytes from the registers

int temperature = ((dataReg[0]<<8 | dataReg[1]) >> 4);

return temperature*0.0625;
}
```

Figure 7: Module of how the temperature was sent to the Galileo Board.

Section 8: Trouble Shooting /1 points

Issue 1: Saving the picture files to the Galileo:

When we first began trying to take a picture with the webcam and saving them to the Galileo, We ran into a problem that each picture we were taking was overwriting the previous picture. We were able to take multiple pictures, but after each picture we were taking, the previous picture would be overwritten. After debugging the main module of the code, it was found that the system used to name the images needed to update prior to saving each image. Figure 8 shows how this fixed in our implementation of the code.

```
date = time(NULL);
cdate = asctime(localtime(&date));
if (temp_Reading > temp_Thresh)
{
        for (int i = 0; i < 25; i++)
        {
                if (cdate[i] == 32)
                {
                        cdate[i] = '_';
                }
                else if (cdate[i] == 58)
                {
                        cdate[i] = '_';
                }
                else if (cdate[i] == 10)
                {
                        cdate[i] = '_';
                }
        }
printf("Taking picture: %s \n", cdate);
        capture_and_save_image(cdate);
```

Figure 8: Code used to name each photograph.

Issue 2: Error in saving each file based on cdate:

As shown in Figure 8, a loop needed to be used to go through the string cdate. Prior to doing this loop, the Galileo was unable to save any of the images to itself as there were illegal characters in the file names due to the naming of the asctime() function. In order to remedy this, the ASCI values of each of the illegal characters were discovered and replaced with an underscore to allow the files to be saved properly to Galileo.

Section 9: Results /0.5 points

When looking at the results in this lab, there were not many results that could be gathered. The whole purpose of this lab was to establish an I2C communication between the temperature sensor and the Galileo and also saving images taken by the webcam to the Galileo. Figure 9 shows the I2C detect output which shows a connection at 0x48, which is where the Temperature sensor was connected.

```
root@galileo:~/Lab3# i2cdetect -y -r 0
     0  1  2  3  4  5  6  7  8  9  a  b  c  d  e  f
00:          -- -- -- -- -- -- -- -- -- -- -- -- --
10: -- -- -- -- -- -- -- -- -- -- -- -- -- -- -- --
20: -- -- -- -- -- UU UU UU -- -- -- -- -- -- -- --
30: -- -- -- -- -- -- -- -- -- -- -- -- -- -- -- --
40: -- -- -- -- -- -- -- -- UU 48 -- -- -- -- -- --
50: -- -- -- -- UU UU UU UU -- -- -- -- -- -- -- --
60: -- -- -- -- -- -- -- -- -- -- -- -- -- -- -- --
70: 70 -- -- -- -- -- -- --
root@galileo:~/Lab3#
```

Figure 9: I2C detect output

Figure 10: Sample image taken by the Webcam.

Shown in Figure 10 is a sample screenshot from the webcam after the temperature sensor read a temperature value over 30 degrees Celsius.

Section 10: Appendix

A1. Main module used in this lab:

```
int
main()
	{
		int cmd;
		double temp_Reading;
		double temp_Thresh = 30;
		//init_temp_sensor();
		time_t date = time(NULL);
		char* cdate;
		while (1)
		{
			printf("\nEnter the Command you would like to do (1 temp sensor-
	triggered pic, 2 take a pic, 3 exit)\n");
			scanf("%d", &cmd);
			if (cmd == 1)
			{
				temp_Reading = get_temp();
				printf("Temperature:%lf degrees C\n", temp_Reading);
				date = time(NULL);
				cdate = asctime(localtime(&date));
				if (temp_Reading > temp_Thresh)
				{
					for (int i = 0; i < 25; i++)
					{
						if (cdate[i] == 32)
						{
							cdate[i] = '_';
						}
						else if (cdate[i] == 58)
						{
							cdate[i] = '_';
						}
						else if (cdate[i] == 10)
						{
							cdate[i] = '_';
```

```
                                        }
                                    }
                printf("Taking picture: %s \n", cdate);
                                    capture_and_save_image(cdate);
                                } else {
                printf("Not hot enough for picture -- \n\tneeds to be greater than %f\n",
        temp_Thresh);
              }
                        }
```

A2: Camera module used in this lab:

```
#include
"opencv2/opencv.hpp"


                        using namespace cv;
                        using namespace std;


                        //read image from filename
                        //if successful, save as a file -- .png and retrun true
                        //if not, return false.
                        bool capture_and_save_image(char* filename)
                        {
                          //init webcam on video0 interface
                          VideoCapture ourCam;
                          ourCam.open(0);


                          Mat image;
                          bool camIsThere = false;


                          //read a frame from the vid camera into image
                          camIsThere = ourCam.read(image);


                          //test if camera not connected
                          if(!camIsThere)
                            return false;
                          //test if a blank image was grabbed
                          if(image.empty())
                            return false;
```

```
        //make jpeg format parameters
        vector<int> compression_params;
        compression_params.push_back(IMWRITE_JPEG_QUALITY);
        compression_params.push_back(95);


        //make filename
        string fn = string(filename) + string(".jpeg");

        //Code from inWrite() example in OpenCV docs
        //try to save to file
        try{
          imwrite(fn, image, compression_params);
        }
        catch (Exception& ex) {
          cout << "Issue saving file: " << ex.what() << endl;
          return false;
        }


        return true;
      }
```

A3: Temperature sensor code used in this lab:

```
#ifndef
TEMP_SENSOR_H
        #define TEMP_SENSOR_H




        #include "mraa.hpp"


        #define TEMPERATURE_REGISTER 0x00    //According to table 1 in the TMP102
        Datasheet
        #define CONFIG_REGISTER 0x01         //These values will determine what
        register we are
        #define T_LOW_REGISTER 0x02                 //communicating with..not sure
        if we them.....
        #define T_HIGH_REGISTER 0x03
```

```
#define TMP102Address 0x48
using namespace mraa;



double get_temp(); // temperature in C


double get_temp(){
        I2c i2c(0);
        i2c.address(TMP102Address);


uint8_t dataReg [2];


int buffer = i2c.read(dataReg,2); // read two bytes from the registers


int temperature = ((dataReg[0]<<8 | dataReg[1]) >> 4);


return temperature*0.0625;
}
#endif
```