Section 1: General Lab Info /0.5 points

EECE.4800 Microprocessors II and Embedded System Design

11024

Lab 3: Controlling an I^{2}C Device

Professor Yan Luo

Group number: 1

Jose Velis

November 16, 2017

November 20, 2017

Section 2: Contributions /1 points

1. Group Member 1 – Jose Velis

   Responsible for temperature sensor I2C implementation. Light troubleshooting and finding the temperature threshold for camera activation.

2. Group Member 2 – Grayson Colwell

   Responsible for main program logic implementation and some camera code (date/time). Troubleshooting. Setting up Galileo for file transfers and compiling.

3. Group Member 3 – Andy MacGregor

   Responsible for implementing opencv library and the rest of camera code. Some help with temperature sensor implementation. This consisted of clearing up how the mraa library works and what functions do. Troubleshooting and setting up Galileo for file transfers and compiling.

Section 3: Purpose /0.5 points

The purpose of the lab was to implement a temperature sensor using the I^{2}C communications protocol. The tempearute sensor measures the current temperature an stores it in a set of registers. These registers are then accessed by the Galileo using the protocol and then displayed to the user. The project also consisted of having a camera that could take pictures when commanded to do so, but also when certain temperature conditions were met. The purpose of having the camera was to implement the opencv library.

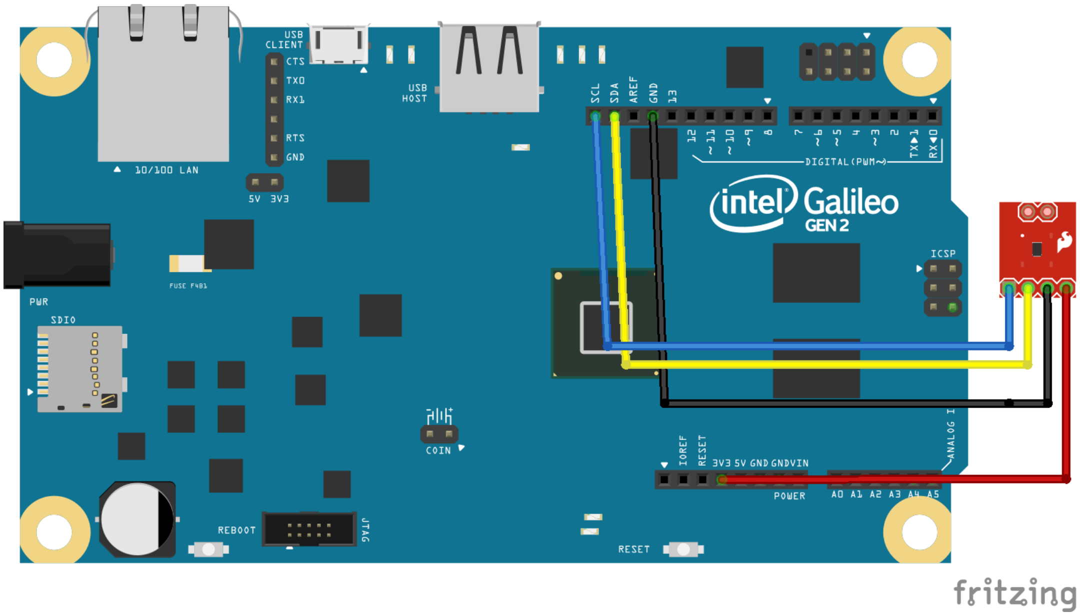
Section 4: Introduction /0.5 points

In this lab, I²C was used to communicate with a temperature sensor. The TMP102 temperature sensor was connected to the Galileo via the SDA and SCL pins. The device operates at 3.3V. on the Galileo a simple command interface was used to select which function to do. Connected to the Galileo was also a USB camera. The camera software was implemented using the opencv library. This library handles the communications protocol and camera functions.

Section 5: Materials, Devices and Instruments /0.5 points

- FTDI cable, operates at 5V
- Breadboard & wiring
- Lab oscilloscope and power supply
- Intel Galileo Gen 2 embedded computer system
- USB Camera
- TMP102 i2c temperature sensor

Section 6: Schematics /0.5 points



Section 7: Lab Methods and Procedure /2 points

Hardware Design:

This lab was simple in the hardware department. As mentioned above, SDA and SCL were connected to the SDA and SCL input pins on the Galileo and the temperature sensor also got power from the Galileo. The camera was connected to the Galileo via the on-board USB port.

Software Design:

The software was design around the mraa and opencv libraries. The mraa library was used to communicate with the TMP102 temperature sensor and the opencv for the camera. A basic flow chart can be seen in figure 1.

The basic code on the Galileo is as follows:

- Initialize all libraries and their objects
- Start i2c and camera interfaces, this is done through functions in the mraa and opencv libraries.
- Prompt user for a command
- Obtain the temperature or take a picture based on a set temperature threshold

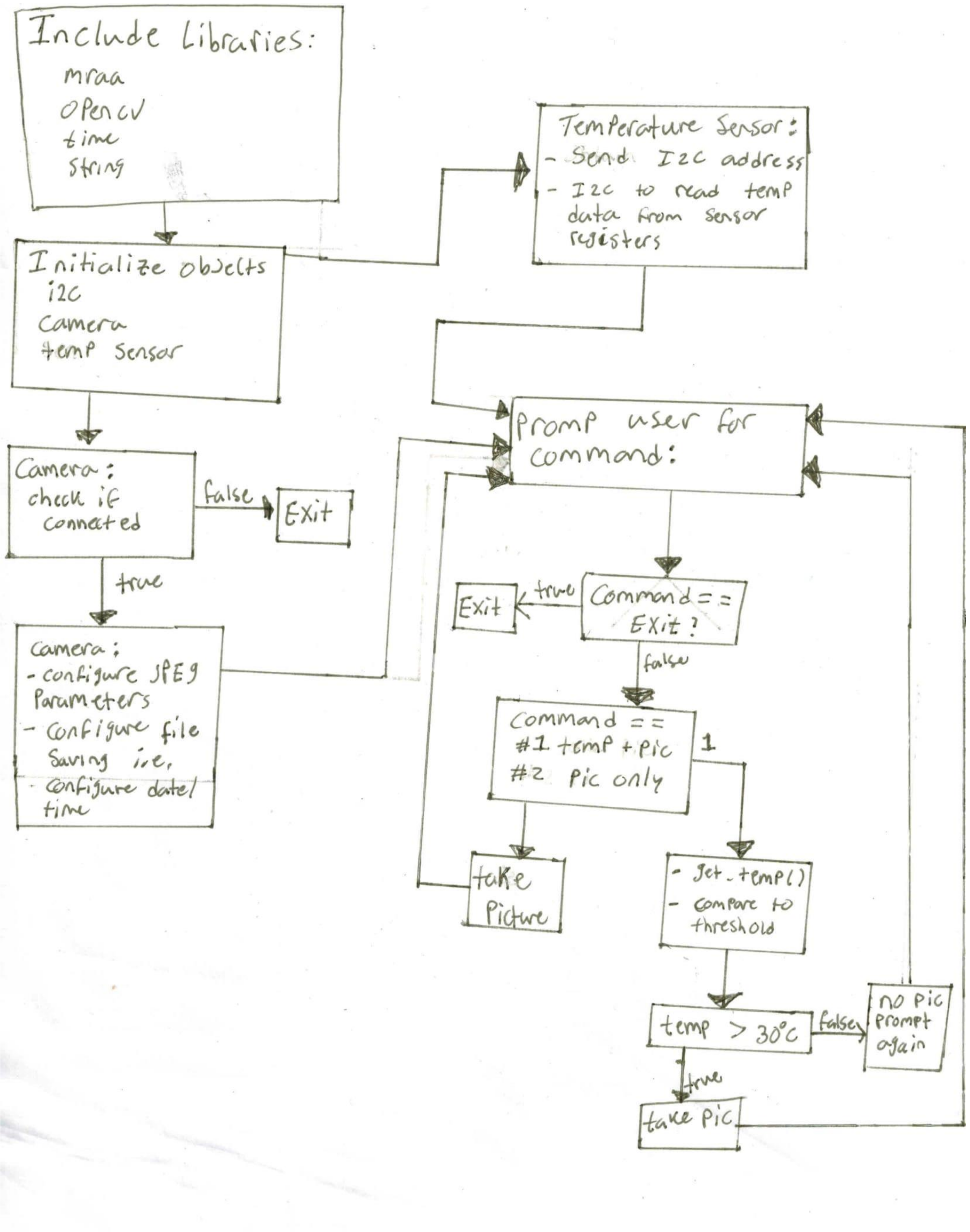


Figure 1 – Software flow chart

Some sample code for the TMP102 sensor is show below:

```
#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
```

Below is a code snippet of the camera file, this code is what initializes the camera and also checks to see if the camera is connected. The code also starts to set the file saving mechanisms:

```
#include
"camera.h"


             #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;
```

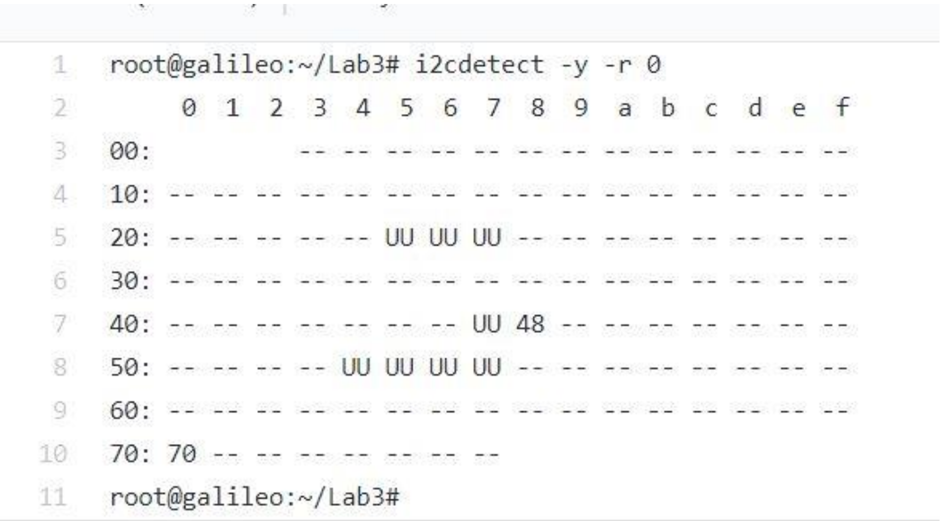
Section 8: Trouble Shooting /1 points

Issue 1: We had issues with correctly shifting the data coming out of the TMP102 for correct temperature display. We solved this by using debugging tools within the terminal to step through each step.

Issue 2: The camera code went relatively straightforward, it was a learning curve with the new librarie but by viewing examples it was implemented.

Section 9: Results /0.5 points

Below is a i2c detect command ran on the Galileo to see if 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#
```

Below is a sample picture taken by the camera: