Lab 3: Building Linux Kernel and Controlling an I2C Device

16.480/552 Microprocessor Design II and Embedded Systems

Instructor: *Prof. Yan Luo*

Group-I

Due:11/28/16

Submitted: 11/28/16

By,

- ***Naga Ganesh Kurapati***
- *Sayali Vaidya*
- *Zubin Pattnaik*
- *Akriti Sharan*

Table of contents

I. Contributions
II. Purpose
III. Introduction
IV. Materials, Devices and Instruments Used
V. Schematics
VI. Lab Methods and Procedure
VII. Trouble Shooting
VIII. Results
IX. Appendix - Code

I. Contributions

Naga Ganesh Kurapati – Worked on configuring the I2C communication between the intel Galileo Gen2 and Gesture sensor APDS-9960. Debugging the codes.

Zubin Patnaik – Worked on designing the circuit, troubleshooting the ports and circuit. Debugging the codes. Configuring the Wi-Fi card on Galileo.

Sayali Vaidya – Worked on configuring the camera to capture picture on Galileo using OpenCV. Debugging the codes.

Akriti Sharan – Worked on configuring the I2C communication between the intel Galileo Gen2 and Temperature sensor TMP102. Debugging the codes.

II. Purpose

The main purpose of this project is to understanding the I2C protocol by configuring I2C devices Gesture sensor(APDS-9960) and Temperature sensor (TMP102) to communicate with intel Galileo Gen2. Using those sensor data to capture an image at threshold value using camera which is configured using OpenCV.

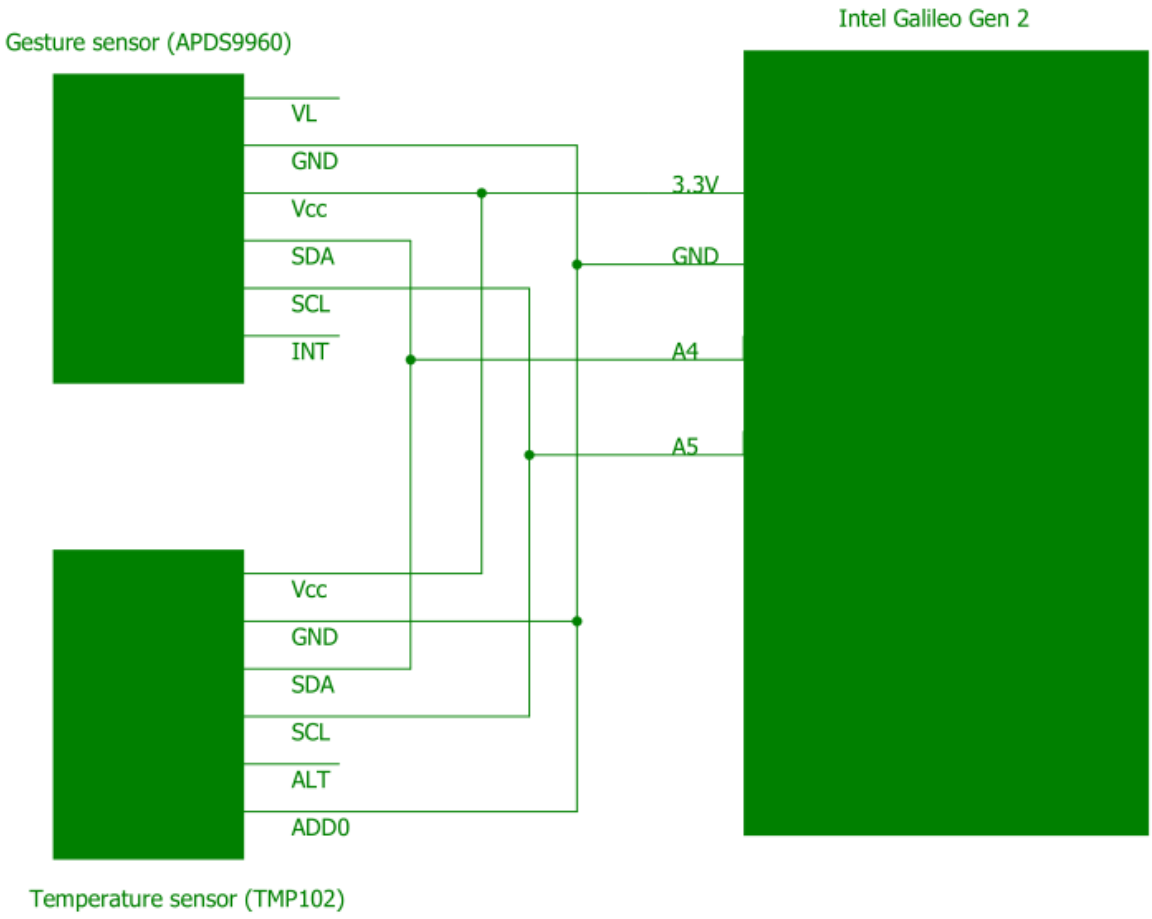
III. Introduction

The main objective of this lab is to read the sensor data from a I2C devices Gesture sensor(APDS-9960) and Temperature sensor (TMP102). Trigger the camera to capture a picture when the required threshold value of the sensor data is reached. Configure the I2C devices communication using the libraries sys/ioctl.h, linux/i2c-dev.h. Configure the camera using OpenCV library to capture the picture. Integrate all this three modules.

IV. Materials, Devices and Instruments Used

1. Bread board
2. Wires to connect
3. Temperature sensor TMP102
4. Gesture sensor APDS-9960
5. two 5k Ohm resistors to connect SCL, SDA to VCC
6. Serial to USB connector
7. Multi-meter
8. Voltage supply (3.3V) from Galileo
9. Intel Galileo Gen 2 Board
10. Yocto Linux
11. Putty Software

V. Schematics



VI. Lab Methods and Procedure

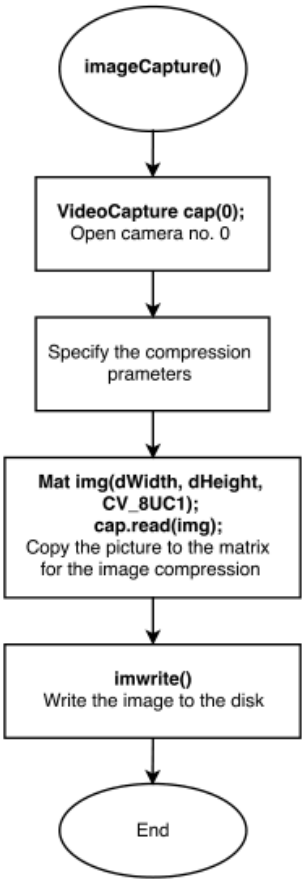
Hardware Design: Galileo is connected to a laptop using serial to USB connector. It is powered from the adaptor cable. I2C bus is designed on the bread board by connecting SCL, SDA pins from the Galileo board and the sensors as shown in the schematic. Those lines are made active high by connected to VCC through 5k Ohm resistors. On Galileo SCL is A5 and SDA is A4. The VCC (3.3) and ground to two sensors is supplied from the Galileo. In this I2C protocol communication Galileo is the master and the two sensors are slaves. The slave address of Gesture sensor APDS-9960 is 0x39 and Temperature sensor TMP102 is 0x48 (by connecting ADD0 to ground selects default address). After the connection, by typing "i2cdetect -r 0" shows all the I2C devices connected to the Galileo as shown in the below picture. Camera is connected to the Galileo board through the USB cable.

```
root@galileo:~#
root@galileo:~#
root@galileo:~# i2cdetect -r 0
WARNING! This program can confuse your I2C bus, cause data loss and worse!
I will probe file /dev/i2c-0 using read byte commands.
I will probe address range 0x03-0x77.
Continue? [Y/n] y
     0  1  2  3  4  5  6  7  8  9  a  b  c  d  e  f
00:          -- -- -- -- -- -- -- -- -- -- -- -- --
10: -- -- -- -- -- -- -- -- -- -- -- -- -- -- -- --
20: -- -- -- -- -- UU UU UU -- -- -- -- -- -- -- --
30: -- -- -- -- -- -- -- -- -- 39 -- -- -- -- -- --
40: -- -- -- -- -- -- -- UU 48 -- -- -- -- -- -- --
50: -- -- -- -- UU UU UU UU -- -- -- -- -- -- -- --
60: -- -- -- -- -- -- -- -- -- -- -- -- -- -- -- --
70: 70 -- -- -- -- -- -- --
root@galileo:~#
```

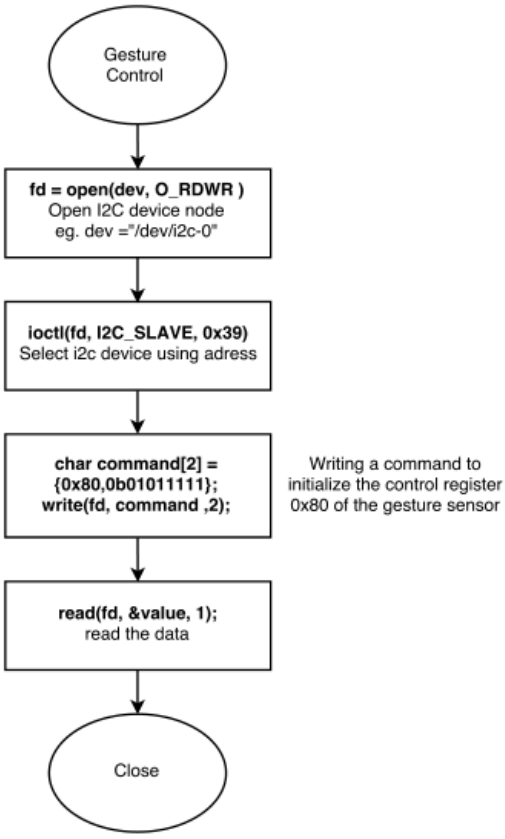
I2C detect on intel Galileo Gen 2

Software Design:

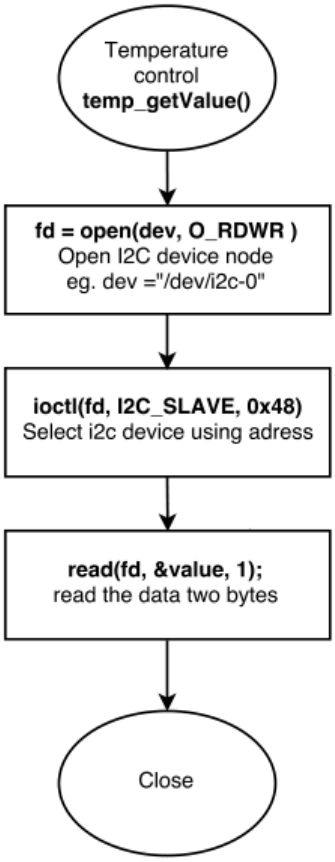
Flow chart for the camera:



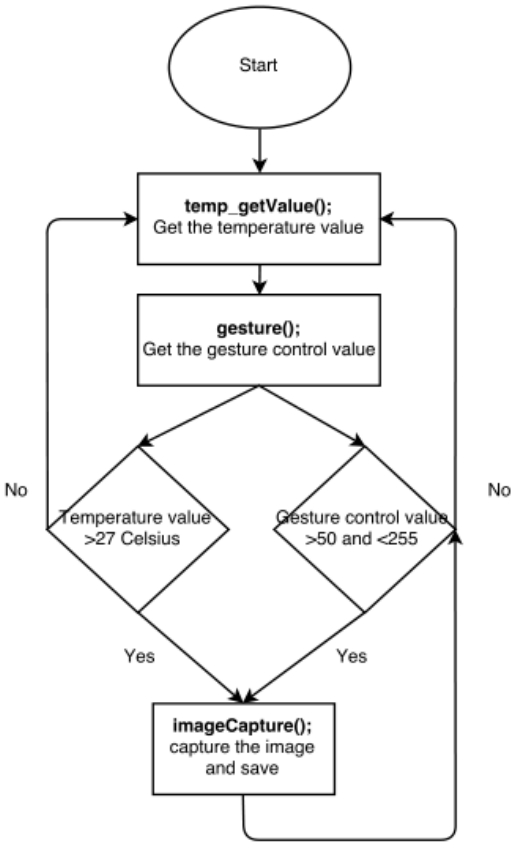
Flow Chart for the gesture:



Flow chat for the temp:



Flowchart of the main program:



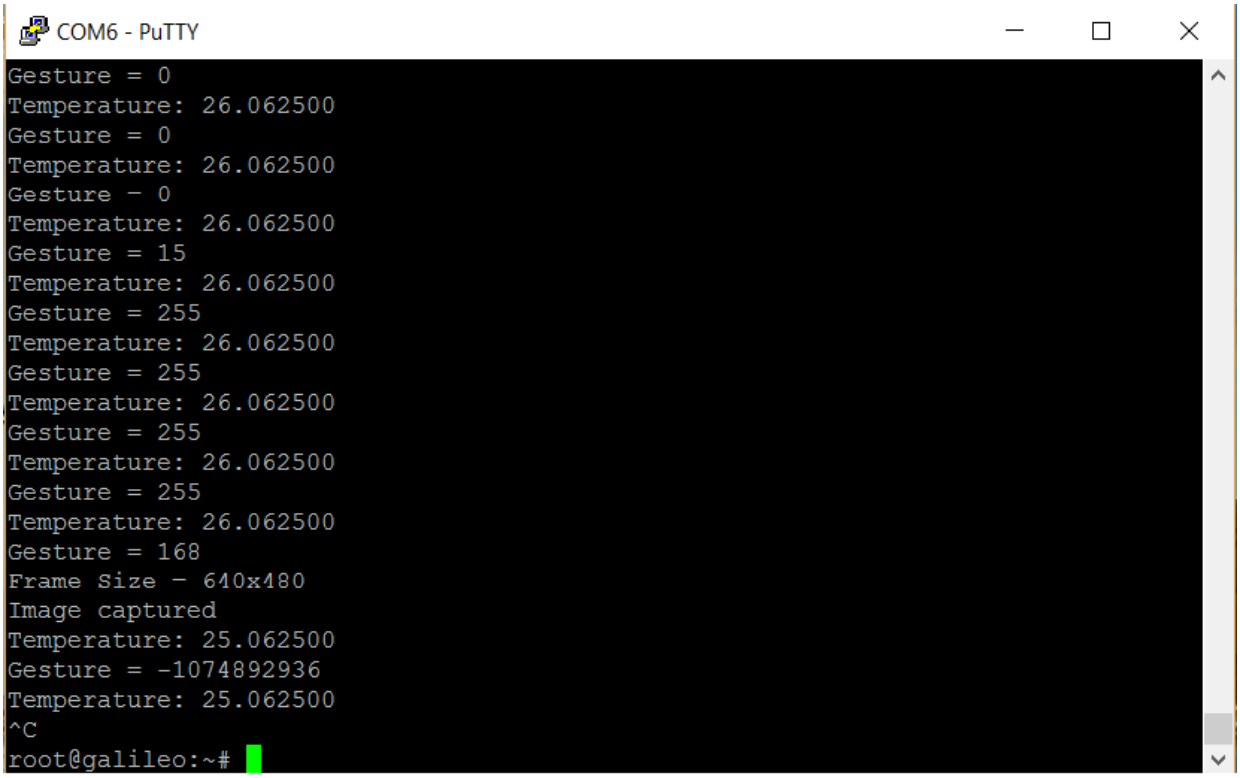
To configure camera to capture image OpenCV library is used and the sensors are configured using i2c device library of the intel Galileo.

Note: Detailed code explanation can be found in the Appendix. Code contains comments.

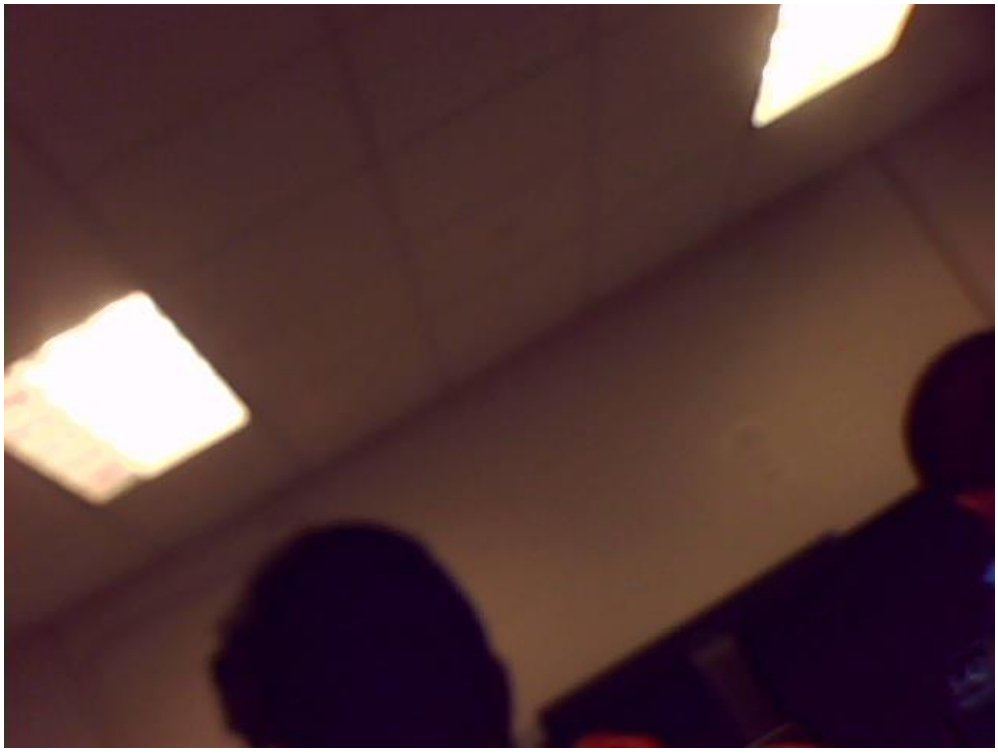
VII. Trouble Shooting

First make sure that supply is 3.3V at each sensor using multi-meter. Using the command “i2cdetect -r 0”, make sure it display all the I2C devices connected to the I2C bus. While running the code, print the temperature on the console and check whether it is varying to the temperature change. Same with gesture sensor, move the hand and see the values varying. Also check camera is triggered at required time by observing the picture captured.

VIII. Results



As you in the above fig., temperature value is 26 in Celsius. The gesture value is variable from 0 to 255 based on the hand distance from the sensor. More value is read when the hand is too near. The camera is triggered if temperature value is greater than 27 or gesture value is between 50 to 255. You can see that image is captured a point where gesture value is 255. The image captured is shown below.



IX. Appendix – Code

1. Makefile: To executables of the program

```
all:lab3

lab3:
	g++ lab3_final.cpp `pkg-config --cflags --libs opencv` -o lab3

clean:
	rm *.o lab3
```

2. gesture.c: To configure Galileo to communicate with gesture sensor

```
int main ()
{
        int i;
        int r;
        int fd;
        unsigned char value[2] ={0,0} ;
        useconds_t delay = 2000;
        //To specify the i2c bus to connect
        char *dev = "/dev/i2c-0";
        //Address of the gesture sensor
        int addr = 0x39;
        //To open the i2c device
        fd = open(dev, O_RDWR );
        //To notify error if cannot open device
        if(fd < 0)
        {
              perror("Opening i2c device node\n");
              return 1;
        }
        //to select the slave device 0x39
        r = ioctl(fd, I2C_SLAVE, addr);
        //Notify if error in selecting the device
        if(r < 0)
        {
               perror("Selecting i2c device\n");
        }
       //Command to be send to configure the sensor
       char command[2] = {0x80,0b01011111};
       //write the command to selected device
       r = write(fd, command ,2);

        if(r != 2)    //Notify error
              printf("ERROR: Writing I2C device\n");
       //Infinite loop
       while(1)
         {
              //To read the gesture control value
              for(i=0;i<2;i++)
              {
```

```
                        value[0] = PDATA;     //Register to be read
                        value[1] = 0x00;      //Value to be read
                        //To read the value
                        r = read(fd, &value[i], 1);

                        //Notify if error in read the value
                        if(r != 1)
                        {
                               perror("reading i2c device\n");
                        }
                        usleep(delay);
                }
                //Open a file to write the sensor value to a text file
                FILE *fp;
                //Open a file
                fp = fopen("gout.txt", "w");
                //write the data to the file
                fprintf(fp,"%d\n", value[1]);
                //Close the file
                fclose(fp);
        }
       //Close i2c device
        close(fd);
        return 0;
}
```

3. temp_getValue() function to read value from temperature sensor

---

```
float temp_getValue()
{
    int i;
    int r;
    int fd;
    float result = 0.0;
    char value[2] ={0} ;
    useconds_t delay = 2000;
    //The i2c bus to be configured
    const char *dev = "/dev/i2c-0";
    //Address of the slave, temperature sensor
    int addr = 0x48;
    //To open i2c device
    fd = open(dev, O_RDWR );
    //Notify if error in opening the device
    if(fd < 0)
    {
        perror("Opening i2c device node\n");
        return 1;
    }
    //To open the slave device with address 0x48
    r = ioctl(fd, I2C_SLAVE, addr);

    //Notify if selecting the device
    if(r < 0)
```

```
    {
        perror("Selecting i2c device\n");
    }
    //To read msb and lsb of the temperature value(12 bits)
    for(i=0;i<2;i++)
    {
            //To read the values, 1 byte at a time
            r = read(fd, &value[i], 1);
            //Notify if error in reading the value
            if(r != 1)
            {
                perror("reading i2c device\n");
            }
            usleep(delay);
    }
    //To combine the two bytes to interpret the data
    float  tlow =0;
    //Data maunipulation
    tlow = (float)(((value[0] << 8) | value[1]) >> 4);
    //Converting to Celsius
    result = 0.0625*(tlow);
    printf("Temperature: %f\n",result);
    //Close the device and return the value
    close(fd);
    return result;
}
```

4. To capture the image using OpenCV

```
void imageCapture()
{
    // open the video camera no. 0
    VideoCapture cap(0);
    // if not success, exit program
    if (!cap.isOpened())
    {
        cout << "ERROR: Cannot open the video file" << endl;

    }
   //get the width of frames of the video
   double dWidth = cap.get(CV_CAP_PROP_FRAME_WIDTH);
   //get the height of frames of the video
   double dHeight = cap.get(CV_CAP_PROP_FRAME_HEIGHT);
   cout << "Frame Size = " << dWidth << "x" << dHeight << endl;
   //vector that stores the compression parameters of the image
   vector<int> compression_params;
   //specify the compression technique
   compression_params.push_back(CV_IMWRITE_JPEG_QUALITY);
   //specify the jpeg quality
   compression_params.push_back(95);
   //Matrix to read the image and compress
   Mat img(dWidth, dHeight, CV_8UC1);
   //Read the image from the camera
   cap.read(img);
```

```
//write the image to file
bool bSuccess = imwrite("/media/card/img.jpg", img, compression_params);
//Notify if error in writing the image
if ( !bSuccess )
   {
        cout << "ERROR : Failed to save the image" << endl;
   }
}
```

5. Gesture function to read the sensor value

```
int gesture()
{
       int result;
       FILE *fp;
       fp = fopen("gout.txt", "r"); //Open the text file
       fscanf(fp, "%d",&result); // read the result from the file
       fclose(fp);                  // Close the file
       printf("Gesture = %d\n", result);
       return (result);             //Return the value
}
```

6. Main function to integrate the all the device to trigger the camera

```
int main(int argc, char* argv[])
{
       //Infinite loop
       while(1)
       {
              //To get temperature value in Celsius
              float temp_result = temp_getValue();
              sleep(1);
              //To get gesture value
              int gesture_result = gesture();
              sleep(1);
              // To trigger camera if temperature is greater than 27
     Celsius and if gesture control value between 50 and 255
              if( temp_result > 27.0 ||
                     (gesture_result >50 && gesture_result < 255))
              {
                     //To capture the image
                     imageCapture();
                     sleep(0.5);
                     printf("Image captured\n");
              }
       }
       return 0;
}
```