

“Course: Microprocessor Design II and Embedded Systems”

“Course # EECE.5520”

“Title - Multithreaded Programming”

“Instructor - Yan Luo”

“Group number - 12”

*Multithreaded Programming*

“Student Name: Aravind Dhulipalla”

“Hand in Date – 12/21/2017”

“Lab Due Date – 12/21/2017”

*Section 2 : Contributions /1 points*

1. Group Member 1 – Aravind Dhulipalla

* Worked on configuring an i2C communication between the intel Galileo and Gesture sensor APDS-9960, HTTP protocol, and multithreaded programming.

• Worked on chip hardware circuit i.e, making connections between the Galileo board, pic micro controller, gesture sensor, temperature sensor.

1. Group Member 2 – Zubair Nadaph

* Worked on configuring the camera to capture picture on Galileo using OpenCV. Debugging the codes, HTTP protocol and multithreaded programming.

• Worked on chip hardware circuit i.e, making connections between the Galileo board, pic micro controller, gesture sensor, temperature sensor.

1. Group Member 3 – Dushyanth Kadari

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

• Worked on chip hardware circuit i.e, making connections between the Galileo board, pic micro controller, gesture sensor, temperature sensor.

*Section 3: Purpose /0.5 points*

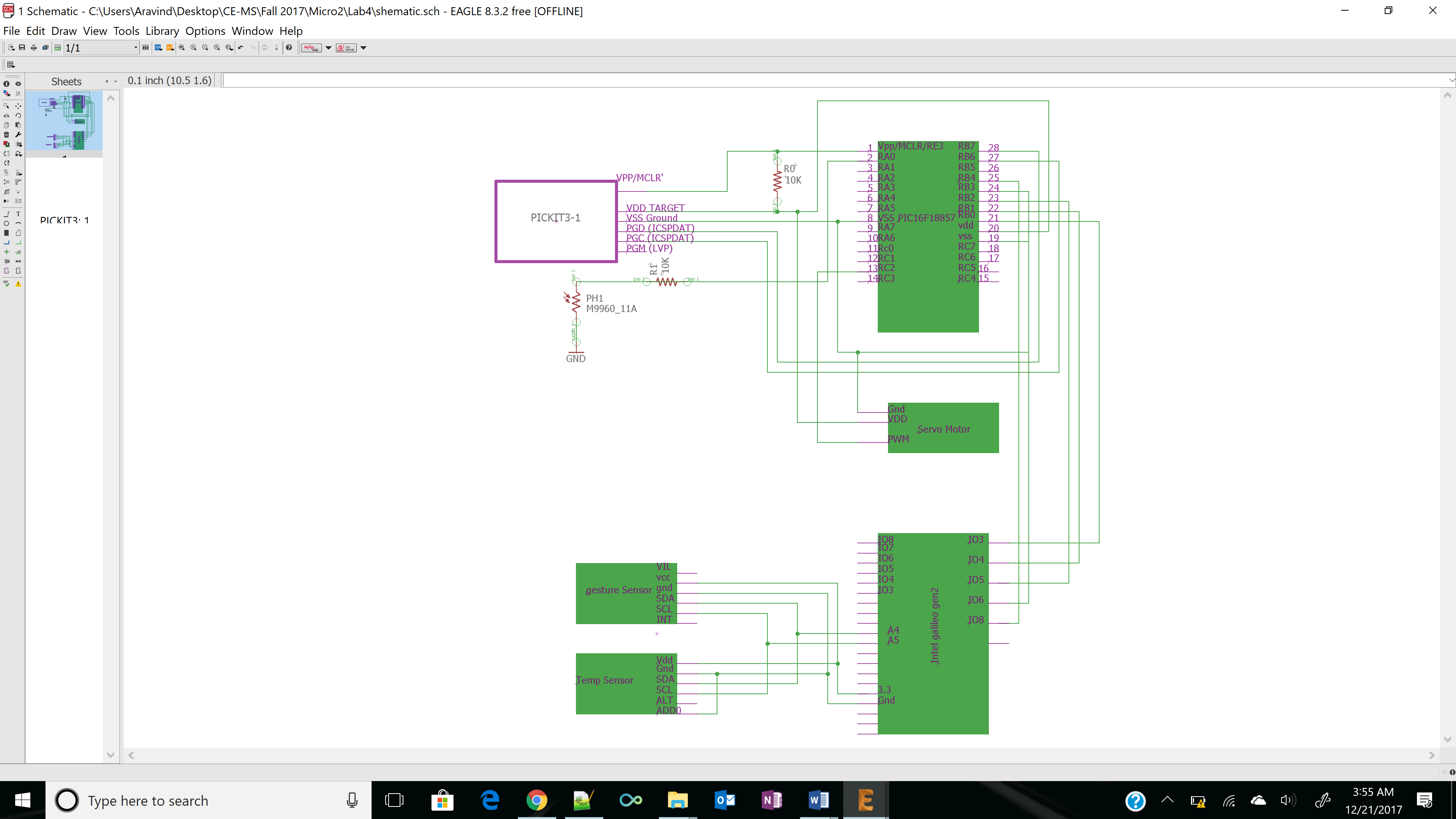
The main purpose of this lab is to understand the multithreading programing using Pthreads. Synchronization of those threads using Mutex. Understanding usage of curl library, HTTP protocol using a client and server application. Understanding of image processing using OpenCV library.

The main objective of this lab is to read the sensor data from a I2C devices Gesture sensor(APDS-9960) and Temperature sensor (TMP102). To read the sensor data (Photo resistor ADC value) from microcontroller PIC16F18857 through strobe communication. Trigger the camera to capture a picture when the required threshold value of the sensor data is reached. Processes the captured image for facial recognition using OpenCV library. And then transfer those images and sensor data to server through HTTP protocol using curl library. Make all these actions concurrent using threads using POSIX thread library.

*Section 5: Materials, Devices and Instruments /0.5 points*

*Section 4: Introduction /0.5 points*

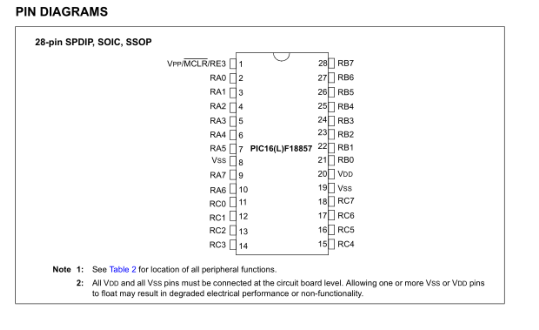
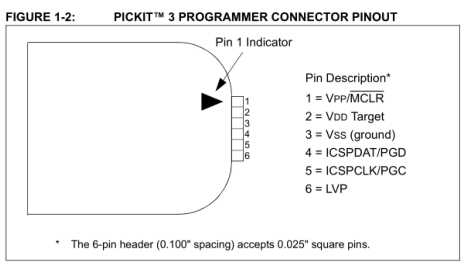
* Bread board
* Wires to connect
* Temperature sensor TMP102
* Gesture sensor APDS-9960
* Serial to USB connector
* Multi-meter
* Voltage supply (3.3V) from Galileo and 5V for servo motor through FTDI
* Intel Galileo Gen 2 Board
* Yocto Linux
* Putty Software
* PIC16F18857 microcontroller
* Resistors 2 (10K ohms)
* Servo Motor
* LDR
* Oscilloscope
* WinSCP to get the image



*Section 6: Schematics /0.5 points*

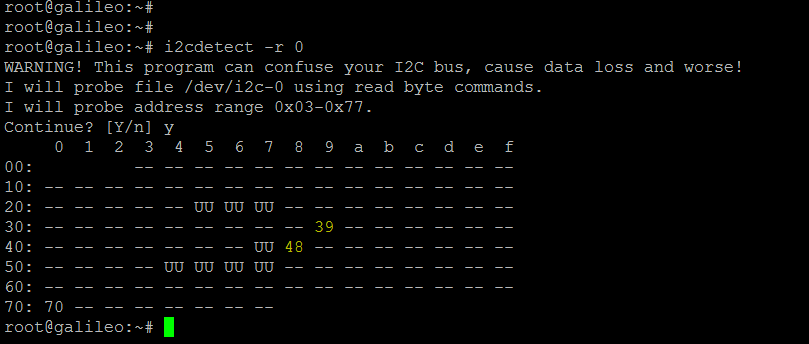
**Hardware design:**

**- PIC Microcontroller:** Initially Pickit3 is connected to the microcontroller. If you observe the pin diagram of both Pickit 3 on top and PIC. Both MCLR, Vdd, Vss, ICSPDAT/PGD, ICSPCLK/PGC are connected to each other. ICSPDAT is pin 27 and ICSPCLK is pin 28 for the PIC. The MCLR is connected to Vdd through 10K ohm resistor. The sensor is connected through ADC Channel 2(Pin 4). And LED is connected to the pin PB0 (Pin21). A 220-ohm resistor is connected in series to the LED, for protection. Pin RB2 is connected to strobe(GPIO8) of Galileo. RC0, RC1, RC2 & RC3 pins are connected to the GPIO3,4,5,6 pins of Intel Galileo.



*Section 7: Lab Methods and Procedure /2 points*

* **I2C devices and camera:** 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.



**Wi-Fi connectivity:** It is configured using connmanctl software, after plugging-in the Wi-Fi card to intel Galileo. Use commands from Yacto linux *connmanctl scan* *wifi* to scan the Wi-Fi networks, *connmanctl servies* to view the Wi-Fi networks and *connmanctl connect* $Wi-Fi-id to connect to the selected Wi-Fi network.

**Software design:**

**Modules used in PIC:**

ADC, PWM, Timer 2 configured accordingly with reference to the data sheet.

**Galileo GPIO’s:**

Gpio’s are initialized by exporting and unexport them when we are done using them.

Thread1:

It performs the following tasks:

1. A set of options to configure the sensors are created for user, it takes the value from user.
2. Performs the actions mentioned in the options
3. Sets flags like update and capture if user selects the option for gesture.

Thread2:

It performs the following tasks:

1. Sets up the apds 9960 geture sensor and programs the bits of enable register.
2. Reading the registers form sensor to get relevant data
3. Checks for gesture and takes picture if beyond threshold.

Thread3:

It performs the following tasks:

1. If the capture flag is set high then enters into it
2. Takes the username/id, ip address, status, adc values, time-date and image and uploads on server.

Mutexes are used whenever we are sharing a data between two threads. The global variables are protected so that no two threads are accessing this variable at the same time. By this the other threads will be able to see if there is any update that is done by any of the threads.

Flow Chart :

A close up of a map

Description generated with high confidence

*Section 8 : Trouble Shooting /1 points*

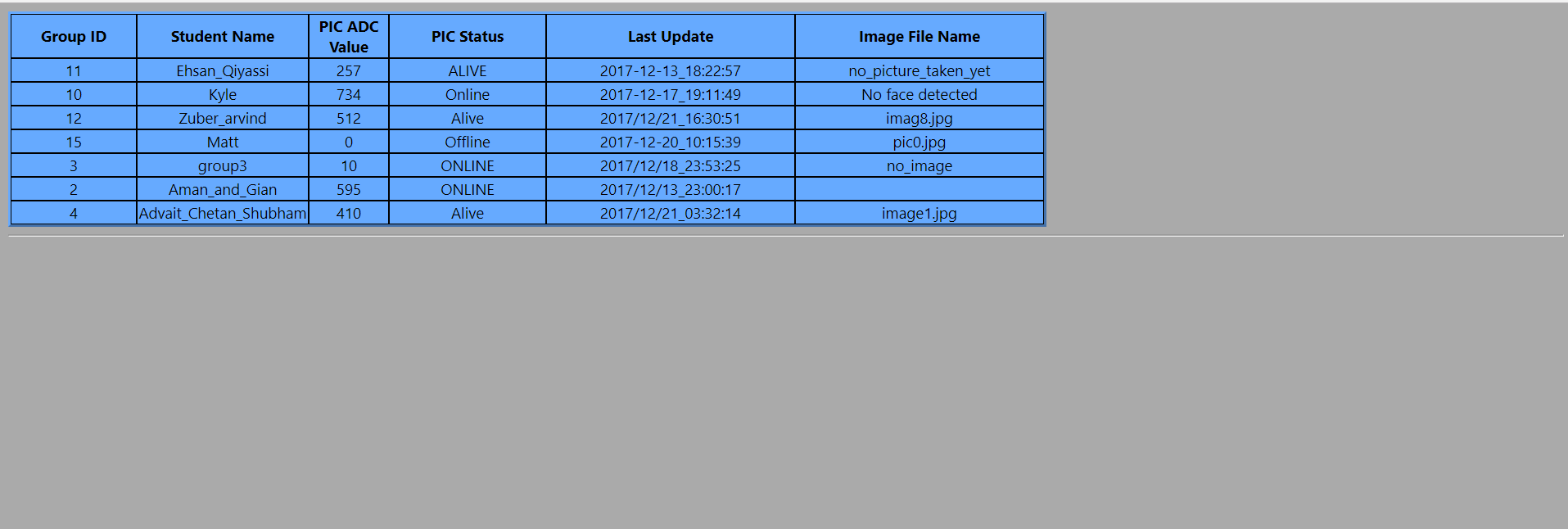
In the first, the response from both the sensors will be responsible for the taking of pictures so, it would be difficult to decide which is responsible. To solve this temperature sensor threshold values are set to high value and can meet it only at certain special conditions and it can also be changed if necessary.

Second Issue was limited availability of the Server most of the time the server was not available almost all the time when we are working. We used that python script which you gave us, and it took us almost 2 hours to figure and run that script.

Third Issue was the HTTP post used by CURL library. We had to figure out that we have to use HTTP Get request instead of Post. It would be good if you had given us the HTTP Get request in the sample code for CURL Library.

*Section 9 : Results /1 points*

As you can see below the PIC data and no image is sent. Our group is 12. PIC value is 512.

**

*The* Image is only taken when there is a UP gesture or the temperature is more than 250 C.

*Section 10: Appendix*

***Code:***

*/\*By*

*Aravind Dhulipalla, Zubair Nadaph, Dushyanth Kadari*

*for Lab assignment 4,EECE. Microprocessors Systems II and Embedded Systems*

*UMASS LOWELL*

*\*/*

*#include <pthread.h>*

*#include <stdio.h>*

*#include <stdlib.h>*

*#include <stdbool.h>*

*#include <curl/curl.h>*

*#include <sys/stat.h>*

*#include <time.h>*

*#include "opencv2/opencv.hpp"*

*#include <iostream>*

*#include <cstdio>*

*#include <fcntl.h>*

*#include <unistd.h>*

*#include <sys/ioctl.h>*

*#include <linux/i2c-dev.h>*

*#define UP 1*

*#define DOWN 2*

*#define LEFT 3*

*#define RIGHT 4*

*#define ACK 0xF*

*using namespace cv;*

*using namespace std;*

*pthread\_mutex\_t mutex,mutex2;*

*int ldrvalue;*

*int update;*

*static int capture=0;*

*char buffer[100];*

*useconds\_t delay = 2000;*

*char \*dev = "/dev/i2c-0";*

*int fd = open(dev, O\_RDWR );*

*/\*-------------------------------------------------------------------------*

*APDS9960\_write()*

*writes the commands to i2-c devices*

*---------------------------------------------------------------------------\*/*

*bool APDS9960\_write(unsigned char address,unsigned char command)*

*{*

*unsigned char command1[2] = {address,command};*

*int r = write(fd,&command1,2);*

*if(r<0)*

*{*

*printf("error wrinting to address: %d",address);*

*return false;*

*}*

*else*

*return true;*

*}*

*void Imagecapture()*

*{*

*VideoCapture cap(0); // open the video camera no. 0*

*if (!cap.isOpened()) // if not success, exit program*

*{*

*cout << "ERROR: Cannot open the video file" << endl;*

*}*

*double dWidth = cap.get(CV\_CAP\_PROP\_FRAME\_WIDTH); //get the width of frames of the video*

*double dHeight = cap.get(CV\_CAP\_PROP\_FRAME\_HEIGHT); //get the height of frames of the video*

*cout << "Frame Size = " << dWidth << "x" << dHeight << endl;*

*vector<int> compression\_params; //vector that stores the compression parameters of the image*

*compression\_params.push\_back(CV\_IMWRITE\_JPEG\_QUALITY); //specify the compression technique*

*compression\_params.push\_back(95); //specify the jpeg quality*

*Mat img(dWidth, dHeight, CV\_8UC1);*

*cap.read(img);*

*static int i =0;*

*snprintf(buffer,100,"Img%d.jpg",i);*

*i++;*

*bool bSuccess = imwrite(buffer, img, compression\_params); //write the image to file*

*if ( !bSuccess )*

*{*

*cout << "ERROR : Failed to save the image" << endl;*

*}*

*}*

*/\*-------------------------------------------------------------------------*

*Read\_gesture()*

*reads the gesture value from the APDS9960 and sense the gesture value and returns*

*the gesture value.*

*---------------------------------------------------------------------------\*/*

*unsigned char read\_gesture()*

*{*

*unsigned char GF4 = 0xAB;*

*unsigned char STATUS = 0x93;*

*unsigned char GFLVL = 0xAE;*

*unsigned char GSTATUS = 0xAF;*

*unsigned char GUP = 0xFC;*

*unsigned char GDOWN = 0xFD;*

*unsigned char GLEFT = 0xFE;*

*unsigned char GRIGHT = 0xFF;*

*unsigned char GF4\_V,STATUS\_V,GFLVL\_V,GSTATUS\_V;*

*unsigned char GUP\_V[32] , GDOWN\_V[32], GLEFT\_V[32] ,GRIGHT\_V[32] ;*

*unsigned char valid\_up[1],valid\_down[1],valid\_left[1],valid\_right[1];*

*while(1)*

*{*

*write(fd,&GF4,1);*

*usleep(delay);*

*read(fd,&GF4\_V,1);*

*//printf("Status : %d\n",GF4\_V);*

*write(fd,&STATUS,1);*

*usleep(delay);*

*read(fd,&STATUS\_V,1);*

*//printf("Status : %d\n",STATUS\_V);*

*write(fd,&GSTATUS,1);*

*usleep(delay);*

*read(fd,&GSTATUS\_V,1);*

*//printf("GSTATUS: %d\n",GSTATUS\_V);*

*unsigned char x = GSTATUS\_V & 0x01;*

*//printf("x = %d",x);*

*unsigned char y = STATUS\_V & 0x02;*

*//printf("y = %d",y);*

*if(((GSTATUS\_V & 0x01) ==1) && ((STATUS\_V & 0x04) == 4 ))*

*{*

*if(!APDS9960\_write(0xAB,0x03))*

*{*

*return false;*

*}*

*sleep(1);*

*//printf("valid\n");*

*write(fd,&GFLVL,1);*

*usleep(delay);*

*read(fd,&GFLVL\_V,1);*

*//printf("GFLVL: %d\n",GFLVL\_V);*

*for(int i=0;i<=GFLVL\_V-1;i++) // for reading the 32 datasets*

*{*

*sleep(0.7);*

*write(fd,&GUP,1);*

*usleep(delay);*

*read(fd,&GUP\_V[i],1);*

*//printf("GUP: %d\n",GUP\_V[i]);*

*write(fd,&GDOWN,1);*

*usleep(delay);*

*read(fd,&GDOWN\_V[i],1);*

*//printf("GDOWN: %d\n",GDOWN\_V[i]);*

*write(fd,&GLEFT,1);*

*usleep(delay);*

*read(fd,&GLEFT\_V[i],1);*

*//printf("GLEFT: %d\n",GLEFT\_V[i]);*

*write(fd,&GRIGHT,1);*

*usleep(delay);*

*read(fd,&GRIGHT\_V[i],1);*

*//printf("GRIGHT: %d\n",GRIGHT\_V[i]);*

*}*

*if(!APDS9960\_write(0xAB,0x00))*

*{*

*return false;*

*}*

*valid\_up[1]={0};*

*valid\_down[1] = {0};*

*valid\_left[1] = {0};*

*valid\_right[1] = {0};*

*for(int j=0;j<GFLVL\_V-1;j++)*

*{*

*if(GUP\_V[j] >50){valid\_up[0] = GUP\_V[j];}*

*if(GDOWN\_V[j] >50){valid\_down[0] = GDOWN\_V[j];}*

*if(GLEFT\_V[j] >50){valid\_left[0] = GLEFT\_V[j];}*

*if(GRIGHT\_V[j] >50){valid\_right[0] = GRIGHT\_V[j];}*

*}*

*if((valid\_up[0] == valid\_down[0]) && (valid\_left[0] == valid\_right[0]) && (valid\_down[0] == valid\_left[0]))*

*{cout << "Give a Gesture please"<<endl;}*

*if((valid\_down[0] < valid\_up[0]) && (valid\_left[0] > valid\_right[0]))*

*{*

*cout << "UP GESTURE DETECTED" << endl;*

*return UP;*

*}*

*if((valid\_down[0] > valid\_up[0]) && (valid\_left[0] > valid\_right[0]))*

*{*

*cout << "Down GESTURE DETECTED" << endl;*

*return DOWN;*

*}*

*if((valid\_down[0] > valid\_up[0]) && (valid\_left[0] < valid\_right[0]))*

*{*

*cout << "Left GESTURE DETECTED" << endl;*

*return LEFT;*

*}*

*if((valid\_down[0] < valid\_up[0]) && (valid\_left[0] < valid\_right[0]))*

*{*

*cout << "Right GESTURE DETECTED" << endl;*

*return RIGHT;*

*}*

*else*

*{*

*cout << "Wrong GESTURE DETECTED Please Try again" << endl;*

*break;*

*}*

*}*

*else*

*{*

*// printf("not valid");*

*write(fd,&GFLVL,1);*

*usleep(delay);*

*read(fd,&GFLVL\_V,1);*

*printf("GFLVL: %d\n",GFLVL\_V);*

*for(int i=1;i<=GFLVL\_V;i++)*

*{*

*write(fd,&GUP,1);*

*usleep(delay);*

*read(fd,&GUP\_V[i],1);*

*write(fd,&GDOWN,1);*

*usleep(delay);*

*read(fd,&GDOWN\_V[i],1);*

*write(fd,&GLEFT,1);*

*usleep(delay);*

*read(fd,&GLEFT\_V[i],1);*

*write(fd,&GRIGHT,1);*

*usleep(delay);*

*read(fd,&GRIGHT\_V[i],1);*

*}*

*if(!APDS9960\_write(0xAB,0x00))*

*{*

*return false;*

*}*

*}*

*}*

*return 0;*

*}*

*/\*------------------------------------------------*

*Gesture Enable function*

*Enables the Gesture sensor required register values*

*--------------------------------------------------\*/*

*bool gesture\_enable()*

*{*

*if(!APDS9960\_write(0xA1,0x00))*

*{*

*return false;*

*}*

*//Config1*

*if(!APDS9960\_write(0xA2,0x00))*

*{*

*return false;*

*}*

*//Config2*

*if(!APDS9960\_write(0xA3,0x41))*

*{*

*return false;*

*}*

*//Up Offstet Register*

*if(!APDS9960\_write(0xA4,0x00))*

*{*

*return false;*

*}*

*//Down offset register*

*if(!APDS9960\_write(0xA5,0x00))*

*{*

*return false;*

*}*

*//Left offset register*

*if(!APDS9960\_write(0xA7,0x00))*

*{*

*return false;*

*}*

*//right offset register*

*if(!APDS9960\_write(0xA9,0x00))*

*{*

*return false;*

*}*

*//Pulse count length*

*if(!APDS9960\_write(0xA6,0x47))*

*{*

*return false;*

*}*

*//cofig3*

*if(!APDS9960\_write(0xAA,0x03))*

*{*

*return false;*

*}*

*//config 4*

*if(!APDS9960\_write(0xAB,0x03))*

*{*

*return false;*

*}*

*//clear interrupts*

*if(!APDS9960\_write(0xE7,0x00))*

*{*

*return false;*

*}*

*return true;*

*}*

*/\*----------------------------------------------------------------*

*Temperature()*

*Reads the temperature value from the sensor and returns the value.*

*------------------------------------------------------------------\*/*

*unsigned char Temperature()*

*{*

*int i;*

*int r;*

*int fd2;*

*float result = 0.0;*

*char value[2] ={0} ;*

*char addr = 0x48;*

*//const char \*dev = "/dev/i2c-0";*

*pthread\_mutex\_lock(&mutex);*

*fd = open(dev, O\_RDWR );*

*if(fd < 0)*

*{*

*perror("Opening i2c device node\n");*

*return 1;*

*}*

*r = ioctl(fd, I2C\_SLAVE, addr);*

*if(r < 0)*

*{*

*perror("Selecting i2c device\n");*

*}*

*for(i=0;i<2;i++)*

*{*

*r = read(fd, &value[i], 1);*

*if(r != 1)*

*{*

*perror("reading i2c device\n");*

*}*

*usleep(delay);*

*}*

*float tlow =0;*

*tlow = (float)(((value[0] << 8) | value[1]) >> 4);*

*result = 0.0625\*(tlow);*

*printf("Temperature: %f\n",result);*

*close(fd);*

*pthread\_mutex\_unlock(&mutex);*

*return result;*

*}*

*void Export()*

*{*

*//export the pin 8 GPIO 40*

*system("echo 40 > /sys/class/gpio/export");*

*//export the pin 7 GPIO 38*

*system("echo 38 > /sys/class/gpio/export");*

*//export pin 6 GPIO 1 and SHIFTER GPIO 20*

*system("echo 1 > /sys/class/gpio/export");*

*system("echo 20 > /sys/class/gpio/export");*

*//export pin 5 GPIO 0 and SHIFTER GPIO 18*

*system("echo 0 > /sys/class/gpio/export");*

*system("echo 18 > /sys/class/gpio/export");*

*//export pin 4 GPIO 6 and SHIFTER GPIO 36*

*system("echo 6 > /sys/class/gpio/export");*

*system("echo 36 > /sys/class/gpio/export");*

*}*

*void UnExport()*

*{*

*//export the pin 8 GPIO 40*

*system("echo 40 > /sys/class/gpio/unexport");*

*//export the pin 7 GPIO 38*

*system("echo 38 > /sys/class/gpio/unexport");*

*//export pin 6 GPIO 1 and SHIFTER GPIO 20*

*system("echo 1 > /sys/class/gpio/unexport");*

*system("echo 20 > /sys/class/gpio/unexport");*

*//export pin 5 GPIO 0 and SHIFTER GPIO 18*

*system("echo 0 > /sys/class/gpio/unexport");*

*system("echo 18 > /sys/class/gpio/unexport");*

*//export pin 4 GPIO 6 and SHIFTER GPIO 36*

*system("echo 6 > /sys/class/gpio/unexport");*

*system("echo 36 > /sys/class/gpio/unexport");*

*}*

*void SetGPIO\_output()*

*{*

*//setting pin8 as an output*

*system("echo out > /sys/class/gpio/gpio40/direction");*

*//Setting pin7 as an output*

*system("echo out > /sys/class/gpio/gpio38/direction");*

*//setting pin6 as an output*

*system("echo out > /sys/class/gpio/gpio1/direction");*

*system("echo out > /sys/class/gpio/gpio20/direction");*

*//setting pin5 as an output*

*system("echo out > /sys/class/gpio/gpio0/direction");*

*system("echo out > /sys/class/gpio/gpio18/direction");*

*//setting pin4 as output*

*system("echo out > /sys/class/gpio/gpio6/direction");*

*system("echo out > /sys/class/gpio/gpio36/direction");*

*}*

*void SetGPIO\_Input()*

*{*

*//Setting pin7 as an input*

*system("echo in > /sys/class/gpio/gpio38/direction");*

*//setting pin6 as an input*

*system("echo in > /sys/class/gpio/gpio1/direction");*

*system("echo in > /sys/class/gpio/gpio20/direction");*

*//setting pin5 as an input*

*system("echo in > /sys/class/gpio/gpio0/direction");*

*system("echo in > /sys/class/gpio/gpio18/direction");*

*//setting pin4 as input*

*system("echo in > /sys/class/gpio/gpio6/direction");*

*system("echo in > /sys/class/gpio/gpio36/direction");*

*}*

*int StrtoInt(char data)*

*{*

*int value;*

*if(data == '0')*

*value =0;*

*if(data == '1')*

*value = 1;*

*return value;*

*}*

*int read\_gpio()*

*{*

*int a;*

*FILE \*fp;*

*system("./gpio\_in.sh 6");*

*fp = fopen("out.txt","r");*

*a = StrtoInt(fgetc(fp));*

*fclose(fp);*

*system("./gpio\_in.sh 0");*

*fp = fopen("out.txt","r");*

*a = a | (StrtoInt(fgetc(fp)) << 1);*

*fclose(fp);*

*system("./gpio\_in.sh 1");*

*fp = fopen("out.txt","r");*

*a = a | (StrtoInt(fgetc(fp))<< 2);*

*fclose(fp);*

*system("./gpio\_in.sh 38");*

*fp = fopen("out.txt","r");*

*a = a| (StrtoInt(fgetc(fp))<<3);*

*fclose(fp);*

*return a;*

*}*

*void \*Interface(void \*Interfaceid)*

*{*

*int cmd,a,adc,data;*

*while(1)*

*{*

*char a = getchar();*

*if(a=='\n')*

*{*

*printf("Enter pressed");*

*pthread\_mutex\_lock(&mutex);*

*update = 1;*

*pthread\_mutex\_unlock(&mutex);*

*printf("Give any one of the command \n 1.Reset 2.Ping 3.PIC LDR VALUE 4.TURN 30 5.TURN 90 6.TURN 120 7.Temperature\n");*

*scanf("%d",&cmd);*

*//make the strobe high*

*switch(cmd)*

*{*

*case 1:*

*Export();*

*SetGPIO\_output();*

*system("echo 1 > /sys/class/gpio/gpio40/value");*

*system("echo 0 > /sys/class/gpio/gpio6/value");*

*system("echo 0 > /sys/class/gpio/gpio0/value");*

*system("echo 0 > /sys/class/gpio/gpio1/value");*

*system("echo 0 > /sys/class/gpio/gpio38/value");*

*usleep(10000);*

*system("echo 0 > /sys/class/gpio/gpio0/value");*

*UnExport();*

*Export();*

*SetGPIO\_Input();*

*system("echo 1 > /sys/class/gpio/gpio40/value");*

*a = read\_gpio();*

*usleep(10000);*

*system("echo 0 > /sys/class/gpio/gpio40/value");*

*UnExport();*

*if(a!=ACK)*

*{*

*printf("pic not available");*

*}*

*break;*

*case 2:*

*Export();*

*SetGPIO\_output();*

*system("echo 1 > /sys/class/gpio/gpio40/value");*

*system("echo 1 > /sys/class/gpio/gpio6/value");*

*system("echo 0 > /sys/class/gpio/gpio0/value");*

*system("echo 0 > /sys/class/gpio/gpio1/value");*

*system("echo 0 > /sys/class/gpio/gpio38/value");*

*usleep(10000);*

*system("echo 0 > /sys/class/gpio/gpio40/value");*

*UnExport();*

*Export();*

*SetGPIO\_Input();*

*system("echo 1 > /sys/class/gpio/gpio40/value");*

*a=read\_gpio();*

*usleep(10000);*

*system("echo 0 > /sys/class/gpio/gpio40/value");*

*UnExport();*

*if(a!=ACK)*

*{*

*printf("pic not available");*

*}*

*break;*

*case 3:*

*Export();*

*SetGPIO\_output();*

*system("echo 1 > /sys/class/gpio/gpio40/value");*

*system("echo 0 > /sys/class/gpio/gpio6/value");*

*system("echo 1 > /sys/class/gpio/gpio0/value");*

*system("echo 0 > /sys/class/gpio/gpio1/value");*

*system("echo 0 > /sys/class/gpio/gpio38/value");*

*usleep(10000);*

*system("echo 0 > /sys/class/gpio/gpio40/value");*

*UnExport();*

*Export();*

*SetGPIO\_Input();*

*system("echo 1 > /sys/class/gpio/gpio40/value");*

*a = read\_gpio();*

*usleep(10000);*

*system("echo 0 > /sys/class/gpio/gpio40/value");*

*if(a==ACK)*

*{*

*system("echo 1 > /sys/class/gpio/gpio40/value");*

*int data = read\_gpio();*

*sleep(0.01);*

*system("echo 0 > /sys/class/gpio/gpio40/value");*

*system("echo 1 > /sys/class/gpio/gpio40/value");*

*data = data | (read\_gpio()<<4);*

*sleep(0.01);*

*system("echo 0 > /sys/class/gpio/gpio40/value");*

*system("echo 1 > /sys/class/gpio/gpio40/value");*

*data = data | (read\_gpio() << 8);*

*sleep(0.01);*

*system("echo 0 > /sys/class/gpio/gpio40/value");*

*UnExport();*

*pthread\_mutex\_lock(&mutex);*

*ldrvalue = data;*

*printf("%d\n",data);*

*pthread\_mutex\_unlock(&mutex);*

*}*

*else*

*{*

*printf("pic not found");*

*update = 0;*

*}*

*break;*

*case 4:*

*Export();*

*SetGPIO\_output();*

*system("echo 1 > /sys/class/gpio/gpio40/value");*

*system("echo 1 > /sys/class/gpio/gpio6/value");*

*system("echo 1 > /sys/class/gpio/gpio0/value");*

*system("echo 0 > /sys/class/gpio/gpio1/value");*

*system("echo 0 > /sys/class/gpio/gpio38/value");*

*usleep(10000);*

*system("echo 0 > /sys/class/gpio/gpio40/value");*

*UnExport();*

*Export();*

*SetGPIO\_Input();*

*system("echo 1 > /sys/class/gpio/gpio40/value");*

*a = read\_gpio();*

*sleep(0.01);*

*system("echo 0 > /sys/class/gpio/gpio40/value");*

*UnExport();*

*break;*

*case 5:*

*Export();*

*SetGPIO\_output();*

*system("echo 1 > /sys/class/gpio/gpio40/value");*

*system("echo 0 > /sys/class/gpio/gpio6/value");*

*system("echo 0 > /sys/class/gpio/gpio0/value");*

*system("echo 1 > /sys/class/gpio/gpio1/value");*

*system("echo 0 > /sys/class/gpio/gpio38/value");*

*usleep(10000);*

*system("echo 0 > /sys/class/gpio/gpio40/value");*

*UnExport();*

*Export();*

*SetGPIO\_Input();*

*system("echo 1 > /sys/class/gpio/gpio40/value");*

*a = read\_gpio();*

*usleep(10000);*

*system("echo 0 > /sys/class/gpio/gpio40/value");*

*UnExport();*

*if(a!=ACK)*

*{*

*printf("pic not ready");*

*}*

*break;*

*case 6:*

*Export();*

*SetGPIO\_output();*

*system("echo 1 > /sys/class/gpio/gpio40/value");*

*system("echo 1 > /sys/class/gpio/gpio6/value");*

*system("echo 0 > /sys/class/gpio/gpio0/value");*

*system("echo 1 > /sys/class/gpio/gpio1/value");*

*system("echo 0 > /sys/class/gpio/gpio38/value");*

*usleep(10000);*

*system("echo 0 > /sys/class/gpio/gpio40/value");*

*UnExport();*

*Export();*

*SetGPIO\_Input();*

*system("echo 1 > /sys/class/gpio/gpio40/value");*

*a = read\_gpio();*

*usleep(10000);*

*system("echo 0 > /sys/class/gpio/gpio40/value");*

*UnExport();*

*if(a!=ACK)*

*{*

*printf("pic not ready");*

*}*

*break;*

*case 7:*

*unsigned char Temp = Temperature();*

*break;*

*}*

*sleep(2);*

*}*

*//create thread 1 & 2*

*}*

*}*

*void \*Sensors(void \*Sensorsid)*

*{*

*while(1)*

*{*

*pthread\_mutex\_lock(&mutex);*

*int cmd = update;*

*pthread\_mutex\_unlock(&mutex);*

*if(update == 0)*

*{*

*unsigned char Temp\_value = Temperature();*

*//char \*dev = "/dev/i2c-0";*

*pthread\_mutex\_lock(&mutex);*

*fd = open(dev, O\_RDWR );*

*int i,r;*

*int addr = 0x39;*

*if(fd < 0)*

*{*

*perror("\nOpening i2c device node\n");*

*}*

*r = ioctl(fd, I2C\_SLAVE, addr);*

*if(r < 0)*

*{*

*perror("\nSelecting i2c device\n");*

*}*

*gesture\_enable();*

*r = APDS9960\_write(0x80,0x4D);*

*if(r<0)*

*{*

*perror("\ngesture engine not started\n");*

*}*

*printf("\ngesture engine started\n");*

*usleep(delay);*

*unsigned char value = read\_gesture();*

*if(!APDS9960\_write(0xAB,0x00))*

*{*

*printf("Error during write to sensor");*

*}*

*if(!APDS9960\_write(0xE7,0x00))*

*{*

*printf("Error during write to sensor");*

*}*

*if(!APDS9960\_write(0x80,0x00))*

*{*

*printf("Error during write to sensor");*

*}*

*if((Temp\_value>20) ||(value == UP))*

*{*

*Imagecapture();*

*cout<<"Gesture Recognised and Picture taken" << endl;*

*pthread\_mutex\_lock(&mutex2);*

*capture = 1;*

*pthread\_mutex\_unlock(&mutex2);*

*}*

*else*

*{*

*cout <<"Gesture Not Correct or Recognised" << endl;*

*}*

*close(fd);*

*pthread\_mutex\_unlock(&mutex);*

*}*

*}*

*}*

*void HTTP\_POST(const char\* url, const char\* image, int size){*

*CURL \*curl;*

*CURLcode res;*

*curl = curl\_easy\_init();*

*if(curl){*

*curl\_easy\_setopt(curl, CURLOPT\_URL, url);*

*curl\_easy\_setopt(curl, CURLOPT\_POST, 1);*

*curl\_easy\_setopt(curl, CURLOPT\_POSTFIELDSIZE,(long) size);*

*curl\_easy\_setopt(curl, CURLOPT\_POSTFIELDS, image);*

*res = curl\_easy\_perform(curl);*

*if(res != CURLE\_OK)*

*fprintf(stderr, "curl\_easy\_perform() failed: %s\n",*

*curl\_easy\_strerror(res));*

*curl\_easy\_cleanup(curl);*

*}*

*}*

*char \*time\_stamp(){*

*char \*timestamp = (char \*)malloc(sizeof(char) \* 16);*

*time\_t ltime;*

*ltime=time(NULL);*

*struct tm \*tm;*

*tm=localtime(&ltime);*

*sprintf(timestamp,"%04d%02d%02d%02d%02d%02d", tm->tm\_year+1900, tm->tm\_mon+1,*

*tm->tm\_mday, tm->tm\_hour-5, tm->tm\_min, tm->tm\_sec);*

*return timestamp;*

*}*

*void \*Client(void \*clientid)*

*{*

*while(1)*

*{*

*if(capture ==1)*

*{*

*printf("sending pic value\n");*

*const char\* hostname="ec2-54-202-113-131.us-west-2.compute.amazonaws.com"; // Server Hostname or IP address*

*const int port=8000; // Server Service Port Number*

*const int id=12;*

*const char\* password="password";*

*const char\* name="Zubair";*

*const int adcval=ldrvalue;*

*const char\* status="HelloAll";*

*const char\* timestamp=time\_stamp();*

*char\* filename="img.jpg"; // captured picture name + incremented file number*

*//fgets(buffer,100,stdin);*

*//filename = (char \*)malloc(strlen(buffer)+1);*

*//strcpy(filename,buffer);*

*char buf[1024];*

*sprintf(buf,"http://%s:%d/update?id=%d&password=%s&name=%s&data=%d&status=%s&timestamp=%s&filename=%s",*

*hostname,*

*port,*

*id,*

*password,*

*name,*

*adcval,*

*status,*

*timestamp,*

*filename);*

*//............*

*// use sprintf() call here to fill out the data "buf":*

*// use the provided URL Protocol in the lab description: replace the "server\_hostname", "portnumber", "var\_xxxx" with the related format specifiers "%d" or "%s"*

*//...................*

*// ========== Don't bother the lines below*

*FILE \*fp;*

*struct stat num;*

*stat(filename, &num);*

*int size = num.st\_size;*

*char \*buffer = (char\*)malloc(size);*

*//fp = fopen(filename,"rb");*

*//int n = fread(buffer, 1, size, fp);*

*// =========== Don't bother the above lines*

*HTTP\_POST(buf, buffer, size);*

*fclose(fp);*

*pthread\_mutex\_lock(&mutex2);*

*capture = 0;*

*pthread\_mutex\_unlock(&mutex2);*

*}*

*}*

*}*

*int main(void)*

*{*

*pthread\_mutex\_init(&mutex,NULL);*

*pthread\_mutex\_init(&mutex2,NULL);*

*pthread\_t thread\_client,thread\_Interface,thread\_Sensors;*

*pthread\_create(&thread\_Interface,NULL,Interface,NULL);*

*pthread\_create(&thread\_Sensors,NULL,Sensors,NULL);*

*sleep(0.01);*

*pthread\_create(&thread\_client,NULL,Client,NULL);*

*pthread\_join(thread\_Interface,NULL);*

*pthread\_join(thread\_Sensors,NULL);*

*pthread\_join(thread\_client,NULL);*

*pthread\_mutex\_destroy(&mutex);*

*pthread\_mutex\_destroy(&mutex2);*

*return 0;*

*}*