**LAB3. Controlling An I2C Device**



Course: Microprocessors Design II and Embedded Systems

Course #: EECE 4800/5520

Instructor: Yan Luo

Group #: 12

Student name: Aravind Dhulipalla

Hand in Date: 11/20/2017

Lab Due Date: 11/20/2017

***Contributions:***

**Team Member 1 – Aravind Dhulipalla**

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

**Team Member 2 – Zubair Nadaph**

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

**Team Member 3 – Dhushyanth Kadari**

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

***Purpose:***

The main purpose of this project is to use the I2C protocol by configuring I2C devices Gesture sensor(APDS-9960) and Temperature sensor (TMP102) to communicate with intel Galileo Gen2. Using these triggers, the camera to take a picture configured through opencv

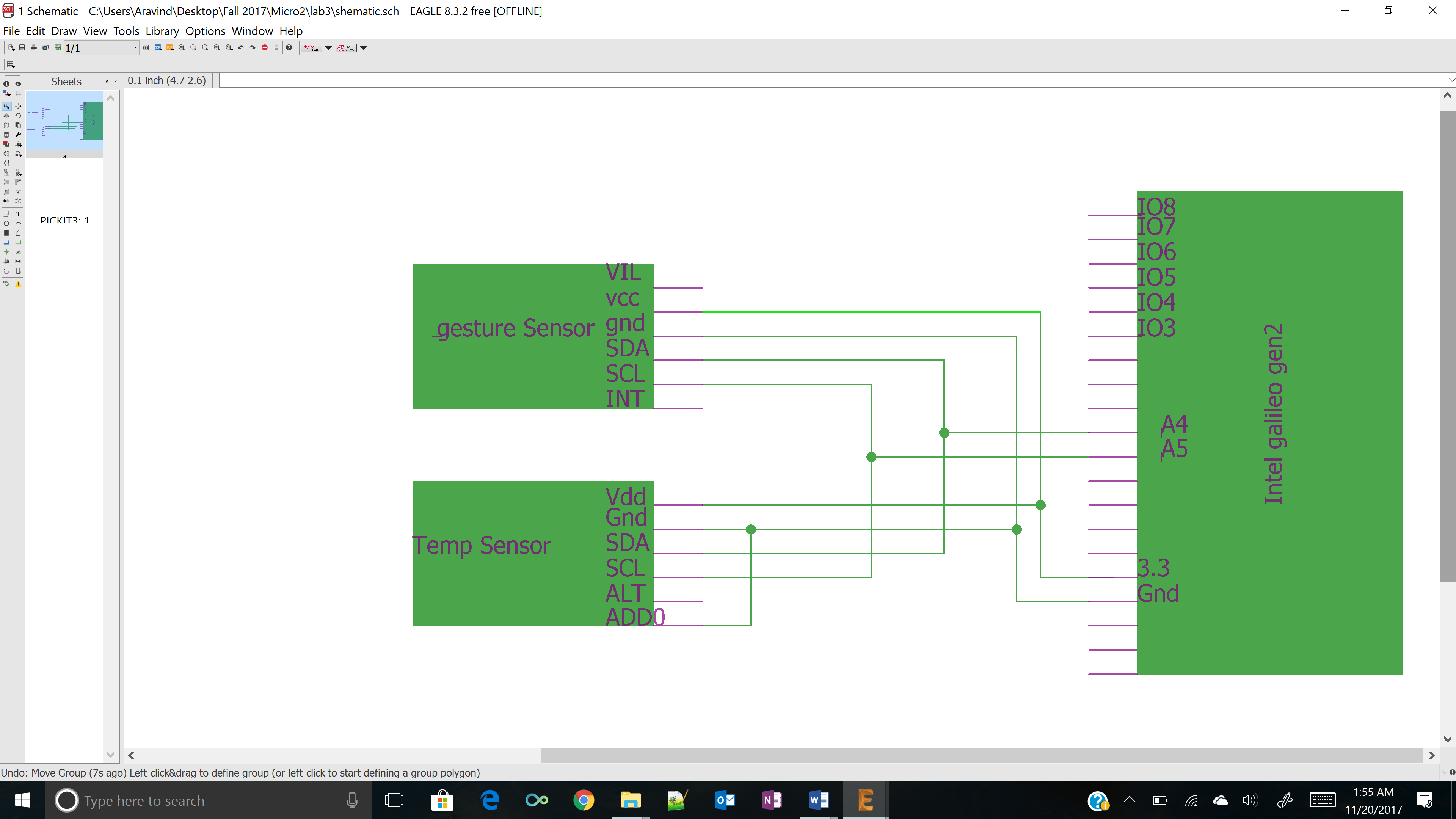
***Introduction:***

The main objective of this project is to interface the Gesture and Temperature with the intel Galileo and communicate between the devices using I2C communication. A threshold value is given and when the sensor data is more than the threshold it will take a picture from the camera. OpenCV is used to capture the image.

***Materials, Devices and Instruments:***

1. Intel Galileo Gen 2
2. Temperature Sensor – TMP102
3. Gesture Sensor – APDS – 9960
4. Serial to USB connector (FTDI cable)
5. Yocto Linux
6. Putty Software
7. Winscp Software
8. Camera
9. Jumper Cables
10. Breadboard.

***Schematics:***



***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.

**Software Design:**

**A close up of a map

Description generated with high confidence**

***Troubleshooting :***

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.

***Results :***

When the Temperature value is greater than 25 or when the gesture value is up the camera picture is taken and saved in the media card. The image shown below is a picture taken when the Gesture value is UP.

A person standing in a dark room

Description generated with very high confidence

***code :***

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

Uses open cv to take the image and saves it in the media card/img

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

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);

bool bSuccess = imwrite("/media/card/img.jpg", img, compression\_params); //write the image to file

if ( !bSuccess )

{

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

}

}

/\*-------------------readgesture()---------------------------------------

Reads the values of UP,Down, Left and Right from the gesture sensor and returns

appropriate gesture

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

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;

}

}

}

}

/\*-------------------------------gesture\_enable()--------------------------

writes the appropriate values for the gesture sensor.(configures)

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

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 Offste 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 value from temperature 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";

fd2 = open(dev, O\_RDWR );

if(fd < 0)

{

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

return 1;

}

r = ioctl(fd2, I2C\_SLAVE, addr);

if(r < 0)

{

perror("Selecting i2c device\n");

}

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

{

r = read(fd2, &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(fd2);

return result;

}

//---------------Main Function----------------------------------------

int main(int argc, const char\*\* argv)

{

int i;

int r;

unsigned char Temp\_value = Temperature();

int addr = 0x39;

if(fd < 0)

{

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

return 1;

}

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

if(r < 0)

{

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

return 1;

}

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))

{

return 1;

}

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

{

return false;

}

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

{

return 1;

}

if((Temp\_value>23) ||(value == UP))

{

Imagecapture();

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

}

else

{

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

}

close(fd);

return(0);

}