Experiment No.2

Motion Sensing System Implementation using raspberry Pi

By: Anand Baraguru Venkateshmurthy (A20429060)

Instructor: Dr. Won-Jae Yi

ECE 442

Lab Date: 03-08-2019

Due Date: 03-15-2019

Acknowledgment: I acknowledge all of the work (including figures and codes) belongs to me and/or persons who are referenced.

Signature: Anand BV

I. Introduction

A. Purpose

To understand methods of communication between microcomputer and I/O devices using raspberry pi, data decryption using symmetric key and a client-server UDP connection.

B. Background

A. Raspberry Pi

Raspberry Pi is a single-board computer used for teaching of computer science basics in schools. In lab, Raspberry Pi 3 is used. A Raspberry Pi 3 has CPU, GPU, RAM (1GB), Network, Storage, GPIO and ports. Because Raspberry Pi is furnished with these technologies such as Bluetooth, Wi-Fi and can be reconfigured by programming, it is used extensively.

B. Adafruit ADXL345

ADXL345, a digital-output 3-axis accelerometer is used as sensor for providing acclereration measurements to Raspberry Pi. SPI or I2C is used to communicate with ADXL345. ADXL345 has four sensitivity levels, 7 bit address 0x53 and 5V input regulated to 3.3V.

C. I2C (Inter-Integrated Circuit Bus)

To transfer data between master and slave devices I2C bus is used. It uses only two wires, SCL and SDA. SDA is the data line and SCL is the clock line. Master drives SCL clock line and slaves respond to master. Slave will never initiate a transmission. Master initiates a transmission by issuing start sequence on I2C bus and stop sequence to end the transaction. I2C address are 7 bits and 10 bits.

D. SPI (Serial Peripheral Interface)

SPI uses 3 wires or 4 wires to transfer data between master and slave devices. The 4 wires are SCK, MOSI, MISO and SS. SCK is a clock line, MOSI and MISO are data lines and SS is for Slave Select. SPI uses SS to select slave device. 2 bytes are used for each transaction. One byte for indicating slave address and another is for data.

E. Client-Server Architecture

Server is a powerful computer, printers (print servers), network traffic (network server). Clients are Personal Computer or workstations. In this lab, Raspberry Pi is used as client and PC as a server.

F. Encryption of Electronic Data

To protect the data transmitted over the Internet, data should be encrypted. Because may be chances of data being eavesdropped. In this lab, Advanced Encryption Standard algorithm is used to encrypt the data. Since keys used for encryption and decryption are same, AES is symmetric key encryption algorithm.

II. Procedure and Equipment List

A. Equipment

Equipment

- 1 x Raspberry Pi 3 single-board computer
- 1 x Adafruit ADXL345 triple-axis accelerometer
- 1 x Desktop (as a data storage server)

B. Procedure

A. Implement Real-Time Motion Sensing Device Using I2C

1. Wire Raspberry Pi and ADXL345 and use I2C for communication between the two.
2. Compile the code provided in Discussion 1 on Raspberry Pi.

   Steps to compile the code on Raspberry Pi

   1) On desktop, create new folder and name it as "Project".
   2) Open "terminal" by pressing combination of "cntrl + alt + t" and type the following command:

      Cd~/Desktop/Project

   3) Compile the source code by typing the following command in the terminal

      g++ -g main.cpp –o exe.out
   4) To debug the program use GDB.
   5) Execute the file by typing:

      ./exe.out
   6) Hold "q" to stop the program.
   7) Check the output on the screen and in the "output.txt".

B. Client Server Communication

1. SPI is used for communication between Raspberry Pi and ADXL345.
2. Compile the code provided in Appendix B by modifying it store a time stamp in the output file.
3. To compile the code following code is used

   g++ -g –lwiringPi –lcrypto client.cpp encryption.cpp –o exe.out
4. Data from sensor will be received, decrypted using Crypto package in Python and displayed on the server, a live plot will be generated using Matplotlib package on the server as well.
5. Copy "server.py" to server computer. Open "anaconda prompt" and navigate to the folder and type "python server.py" in anaconda prompt.
6. Modify IP address and port number. UDP port number used is 6000

III. Results and Analysis.

a) Schematics

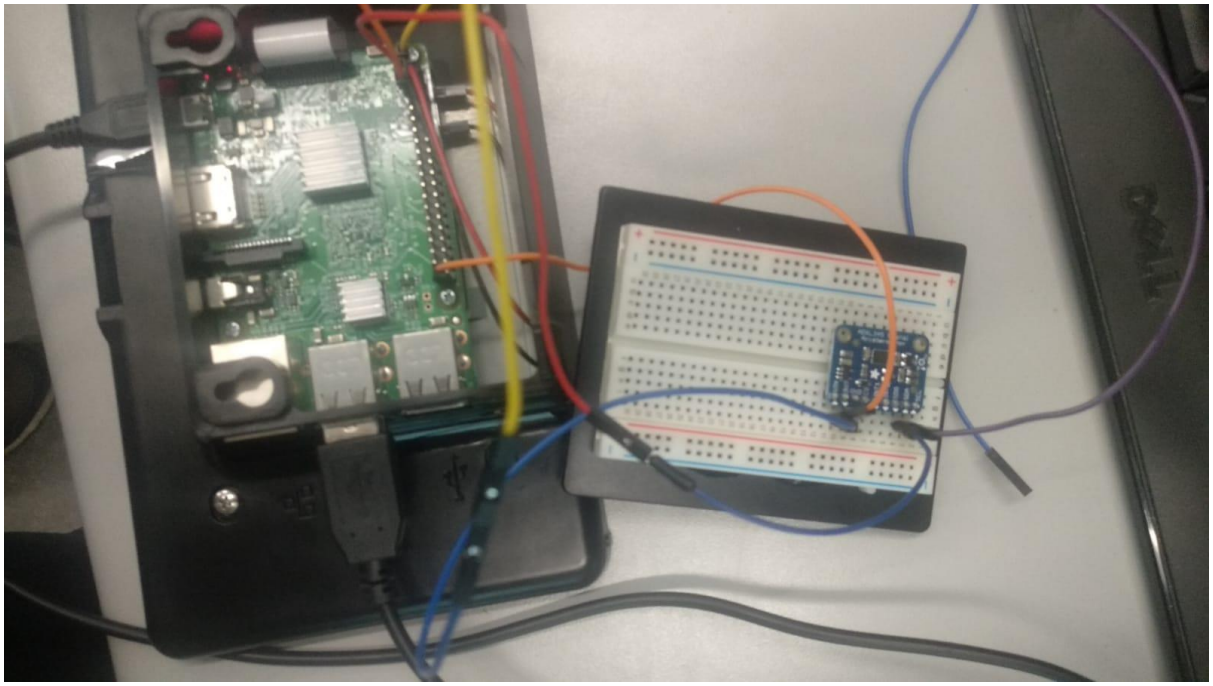


Figure 1. Raspberry Pi Motion-Sensing system using Accelerometer and I2C Bus

Discussion

1) A fully commented code that you implemented in Procedure A.2 (5pts)

```
#include<stdio.h> //Standarad I/O header file
#include<signal.h> //Signal handling header file
#include<sys/time.h> //System/Time header file

#include "i2c-dev.h" //i2c-dev header file
#include "ADXL345.h"//ADXL345 header file

#define I2C_FILE_NAME "dev/i2c-1" // for Rpi B+
void INThandler(int sig); //
int main(int argc, char **argv)//main function
{
   //Open a connection to the I2C userspace control file.
   int i2c_fd = open(I2C_FILE_NAME,O_RDWR); //initializing i2c_fd as integer
   if(i2c_fd <0)
   {
  print("Unable to open i2c control file, err=%d\n",i2c_fd); //print error message unable to
  open i2c control file
                exit(1); //Indicates unsuccessful program termination
   }

   printf("Opened i2c control file, id=%d\n", i2c_fd); // print Opened i2c control file message
   ADXL345 myAcc(i2c_fd); //ADXL345 control file access
   int ret = myAcc.init();
   if (ret)
   {
                printf("failed init ADXL345, ret=%d\n", ret); //print error message failed init
ADXL345
                exit(1);//indicates unsuccessful program termination
   }
   usleep(100 * 1000);// Suspend execution for 100 * 1000 interval
   signal(SIGNT, INThandler);
   short ax, ay, az;// Initialize variables ax, ay and az as short
   //create file IO
   FILE *fp;
   fp = fopen("./output.txt","w+");

   char TimeString[128]; //convert time to string
   timeval curTime; //get time delay
   while(1)
   {
   gettimeoftheday(&curTime, NULL); // get the current time
   strftime(TimeString, B0, "%Y-%m-%d %H:%M:%S",localtime(&curTime.tv_sec);
   printf(TimeString); //print TimeString
   printf(": ");
   myAcc.readXYZ(ax, ay, az); // fetch data from sensor
   printf("Ax : %hi \t Ay : %hi \t Az : %hi \n",ax,ay,az); //print to screen
  printf("----------------------\n");
   fprintf(fp,TimeString); //print to file
   fprintf(fp, ": ");
   fprintf(fp, "Ax : %hi \t Ay : %hi \t Az : %hi \n",ax,ay,az);
   fprintf(fp, "----------------------\n");
```

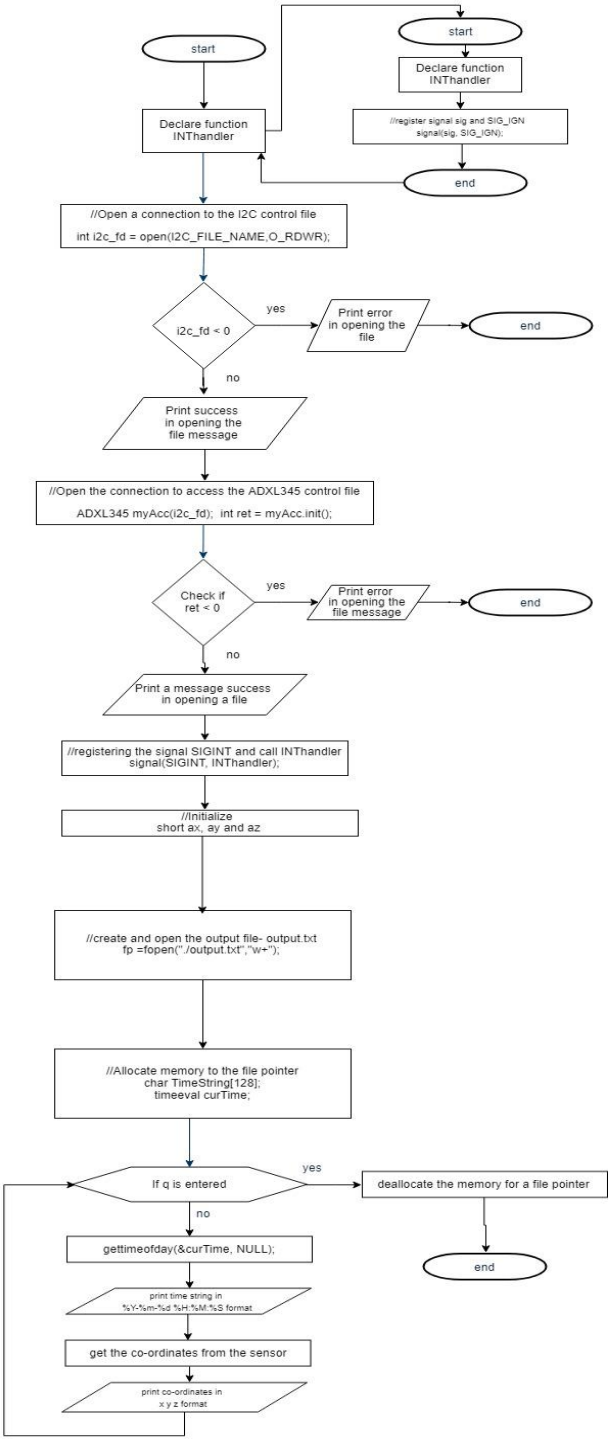
```
    if (getchar() == 'q') break;
    }

    fclose(fp); //deallocate memory for the output file

    return 0;
}

void INThandler(int sig)
{
    signal(sig, SIG_IGN);//signal handling
    exit(0);
}
```

2) Draw a flow chart and explain how the program you implemented in Procedure A.2 works, please elaborate on how Raspberry Pi get the reading from sensors. (5 pts)



3). How does a master device differentiate slave devices using I2C and SPI bus (in terms of protocol hardware implementation)? (10 pts)

Solution: Inter-Integrated Circuit (I2C):

Each I2C device has an unique address of 7 bits or 10 bits. Master device identifies slaves based on their address provided by the manufacturer at the time of manufacture.

Serial Peripheral Interface (SPI):

Multiple slaves can be connected to a master device. SCLK, MOSI and MISO are shared between master and slave devices. But a Chip Select (Slave Select) is a dedicated pin which not shared. Master device uses this Chip Select (Slave Select) to identify different slave devices connected to it.

4).If value 0x0A were found in register 0x31 on ADXL345, what is the behavior of the sensor regarding the register? (10 pts)

Solution:

Register 0x31 DATA_FORMAT (Read/Write)

D7	D6	D5	D4	D3	D2	D1	D0
SELF_TEST	SPI	INT_INVERT	0	FULL_RES	Justify	Range	Range
0	0	0	0	1	0	1	0

- SELF_TEST = 0; disables self-test force
- SPI = 0; sets the device to 4-wire SPI mode
- INT_INVERT = 0; sets the interrupts to active high
- FULL_RES = 1; full resolution mode, where the output resolution increases with the g range set by the range bits to maintain a 4 mg/LSB scale factor
- Justify = 0; right (LSB) justified mode with sign extension
- Range = 10; ± 8g, 10-bit mode

5). What is the maximum sampling rate (samples/second) for the server (PC)? Note each sample comprises acceleration readings in three axes (X, Y, Z). Sampling rate on the server is bottlenecked by following factors: sampling rate of the sensor; FC bus bandwidth; Wi-Fi bandwidth and encoding of data when being transmitted (sent as string or raw floating data). To achieve the theoretical maximum sampling rate on the server how the system need to be configured? (e.g., sampling rate of the sensor, data encoding/decoding PC bus bandwidth, etc.) (20pts)

Solution: The maximum sampling rate of the PC is 44.1 KHz. To achieve the theoretical maximum sampling rate on the server, the sampling rate of the sensor should be 3.2 KHz, I2C bus bandwidth should be 3.2 Mbps.

IV. Conclusions

A motion sensing system using Raspberry Pi 3, Accelerometer, I2C bus and SPI was build. By using UDP connection encrypted data was sent from Raspberry Pi and server and finally data was decrypted and displayed on the screen in a graphical format.

References

[1] Experiment 2 lab manual.

[2] ADXL345 datasheet [online]
(https://www.sparkfun.com/datasheets/Sensors/Accelerometer/ADXL345.pdf)