2014

DESIGN AND IMPLEMENTATION OF HOME USE PORTABLE SMART ELECTRONICS

Liqiang Du
Michigan Technological University

DESIGN AND IMPLEMENTATION OF HOME USE PORTABLE SMART ELECTRONICS

By

Liqiang Du

A REPORT

Submitted in partial fulfillment of the requirements for the degree of

MASTER OF SCIENCE

In Electrical Engineering

MICHIGAN TECHNOLOGICAL UNIVERSITY

2014

This report has been approved in partial fulfillment of the requirements for the Degree of MASTER OF SCIENCE in Electrical Engineering.

Electrical & Computer Engineering

Report Advisor:	*Shiyan Hu*
Committee Member:	*Chaoli Wang*
Committee Member:	*Sumit Paudyal*
Committee Member:	*Zhaohui Wang*
Department Chair:	*Daniel R. Fuhrmann*

Abstract

The widespread of low cost embedded electronics makes it easier to implement the smart devices that can understand either the environment or the user behaviors [3]. The main object of this project is to design and implement home use portable smart electronics, including the portable monitoring device for home and office security and the portable 3D mouse for convenient use. Both devices in this project use the MPU6050 which contains a 3 axis accelerometer and a 3 axis gyroscope to sense the inertial motion of the door or the human hands movement.

For the portable monitoring device for home and office security, MPU6050 is used to sense the door (either home front door or cabinet door) movement through the gyroscope, and Raspberry Pi is then used to process the data it receives from MPU6050, if the data value exceeds the preset threshold, Raspberry Pi would control the USB Webcam to take a picture and then send out an alert email with the picture to the user. The advantage of this device is that it is a small size portable stand-alone device with its own power source, it is easy to implement, really cheap for residential use, and energy efficient with instantaneous alert.

For the 3D mouse, the MPU6050 would use both the accelerometer and gyroscope to sense user hands movement, the data are processed

by MSP430G2553 through a digital smooth filter and a complementary filter, and then the filtered data will pass to the personal computer through the serial COM port. By applying the cursor movement equation in the PC driver, this device can work great as a mouse with acceptable accuracy. Compared to the normal optical mouse we are using, this mouse does not need any working surface, with the use of the smooth and complementary filter, it has certain accuracy for normal use, and it is easy to be extended to a portable mouse as small as a finger ring.

Introduction

Recently, with the fast development of electronic devices and smart home technology, lots of smart devices have been developed for a wide range of applications including wearable devices for convenient use, security electronics considering home security, etc. [8] They are all well designed with the purpose of optimizing the old devices as well as providing a more comfortable life. For example, the surveillance system, it cannot detect whether things happened or not, thus it will have to take videos all the time and will need human resources to monitor in case something really happens [10], therefore, it consumes a lot of power and need additional human resources. In addition, the surveillance can only detect buildings, street, etc, for things as small as a file cabinet, it is totally no use. The other example is the most frequently used device, computer mouse. It significantly changes the experiences of us using the computers, brought us a lot of conveniences. However, it always needs a working surface and will cause the computer mouse fatigue after long time of using [17]. Because of all these weaknesses, some whole new smart devices are quite necessary to optimize them.

In this project, I proposed a portable door monitoring device for home and office security and 3D portable mouse for convenient use.

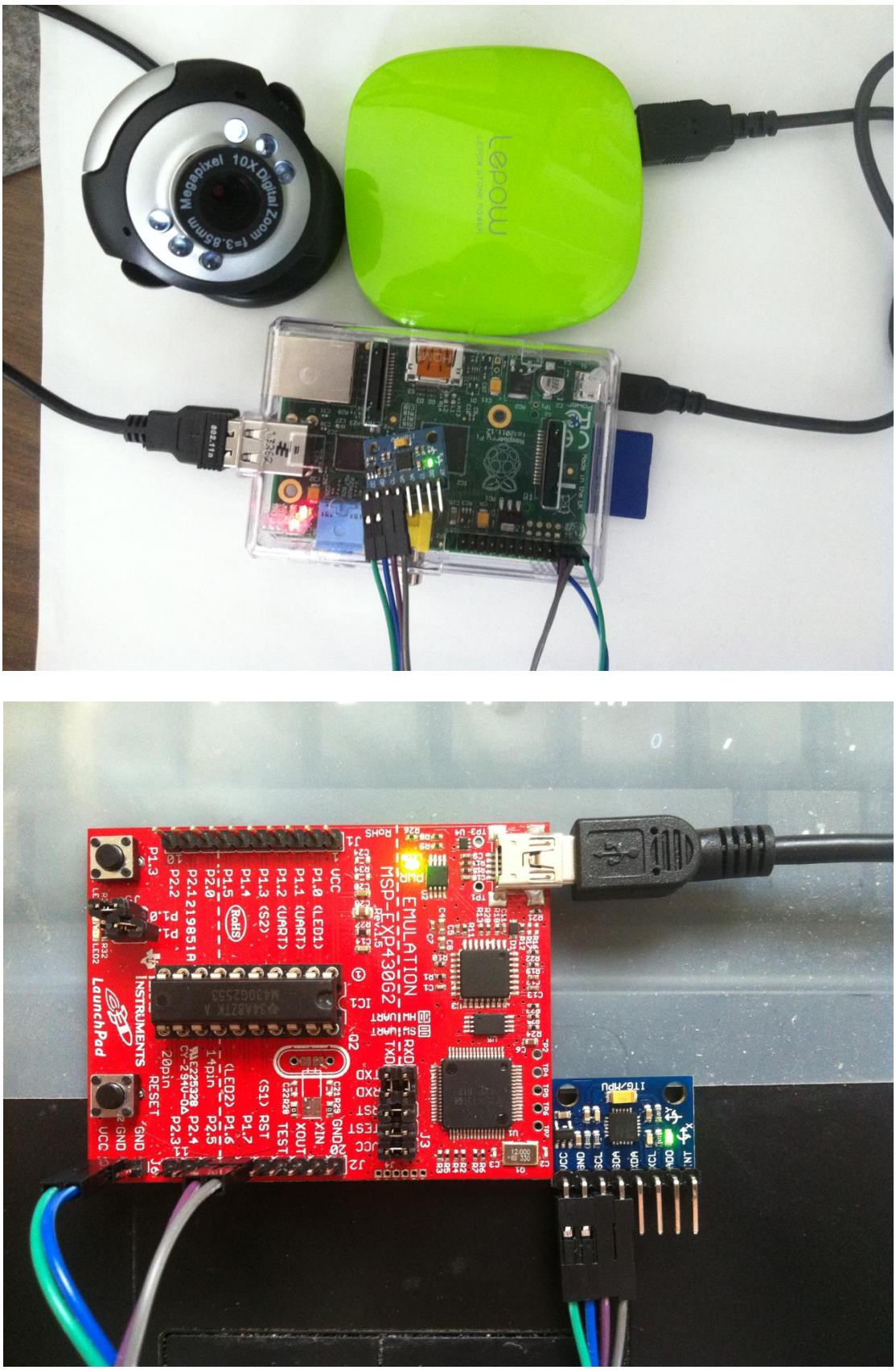

Figure 1 Monitoring device and 3D mouse

The door monitoring device is an event driven system, it will only take pictures or videos when things happened and send out alert email to the users for them to handle the break in immediately. The 3D mouse is using motion sensors to detect hand movement instead of optical sensors, so it does not need any working surface and will be more convenient to use. Both devices are using the MPU6050 to detect motion informations, either for doors or for human hand movement. Figure 1 shows the two system diagram of the two devices. A lot of experiments have been done to determine the threshold value and the hand movement function. Details will provided in the following section.

Part 1 Portable monitoring device for home and office security

1 Introduction

Burglary and theft have always been a headache for ordinary residents, especially for those living in the big cities. According to a report of Crime in the United States 2012, there are in total 2,103,787 burglaries happened across the whole nation. The average loss per burglary is 1675 US dollars. Thus it is quite necessary to find an effective way to significantly reduce it. Surveillance system has always been playing an important role in dealing with the burglary cases. However, it is expensive for ordinary residents to install such kind of system and it is also suffered the defect of not instantaneous, which means it does not inform the user immediately when the burglary happens [14]. What is normally the case is that we can only check the surveillance video after the burglary happened. Furthermore, because of the fact that surveillance systems record all the videos of everyday life, it has to be always in the working mode to record the videos, thus, it is a big waste of the electricity energy. On the other hand, the videos recorded by the surveillance system are in really low definition, sometimes even not clear enough to recognize the theft's face. That is just because if the video is in high quality, the system would be more

expensive and there would be not enough space to store all the video records [11]. Just because all of those weak points of the surveillance system, an energy efficient portable system that can take pictures or videos when the burglary happens and send out an alert signal at the same time is much better than the currently in use surveillance systems. In this project, a portable monitoring device for home and office security is proposed. This device is monitoring through an inertial motion sensor and also a webcam. Instead of taking videos all the time, this device will only take a picture or video through the webcam when something bad happens, thus it significantly reduces a lot of power. In addition, it also has an embedded Linux email server to send out an alert email when burglary or theft happens. Compared to that of the surveillance system, this device is totally portable with its portable external battery; it has really small size that can be placed on any kind of doors, for example, house front door, cabinet door, etc; it is also really cheap that is good for normal residential use. The device is composed of MPU6050, webcam and Raspberry Pi. The MPU6050 is a 3-axis accelerometer and 3-axis gyroscope, the gyroscope is used to detect the movement of the door. The webcam is a normal 640 * 480 resolution camera to take a picture of the theft that breaks in through the door. The Raspberry Pi is an embedded Linux computer to do data processing and send out alert email to the user

when a break in occurs. The whole system is a stand-alone device with its own power source.

2 Methodology

For a device to be used as an effective portable monitoring and alert system, it has to have at least three functions, which are detection, picture taking and alert mechanism [13]. Then first, how can we decide whether someone has been breaking in? According to the data on asecurelif.com, 34% of the burglars enter through the front door and the master bedroom is almost always the first room targeted. Thus if we can detect the movement of the residents' front door or bedroom door, a break in can be easily noticed. Then a movement sensor is needed here. Accelerometer and gyroscope are both good choices. However, if we think deeper about the movement of the door, it is very clear that the door is rotating around the fixed axis, so angular velocity will have a more regular pattern than the acceleration data. Thus, the 3-axis gyroscope in the MPU6050 is used to detect the door movement. After the movement signal is detected, the device need to take a photo of the burglar and sent out an alert signal to the user. A webcam is controlled by Raspberry Pi to take the picture. Finally, what kind of alert information and how can the information be sent out? Text message and email are both really good choices, because text message will need GSM module and SIM card [10], so email is chosen as the alert signal. Normal microcontrollers can control the MPU6050, but they cannot send out alert emails, even with Wi-Fi

module, normal microcontrollers will still need a PC as an email server, which is not a stand-alone system. Therefore, in order to both control the sensor and sent out alert email, embedded Linux would be a really good choice. Thus, Raspberry Pi is chosen as the controller of MPU6050 and also a SMTP email server.

The whole device works in this way, the Raspberry Pi will control the MPU6050 through the I2C bus. When the gyroscope normalized movement signal is detected, Pi will control the webcam to take a picture and send out an alert email along with the photo to the user so that they can handle the emergency immediately.

3 System architecture

The whole system is composed of five parts, the Raspberry Pi embedded Linux controller, the MPU6050 sensor, the Wi-Fi adapter, the webcam and the power supply. Figure 2 shows all the components of the whole system.

3.1Raspberry Pi

3.1.1 Raspberry Pi Introduction

Raspberry Pi is a single board computer with Linux or other small operating systems. It was developed in UK by Raspberry Pi foundation for computer science education use. (wiki-pedia)

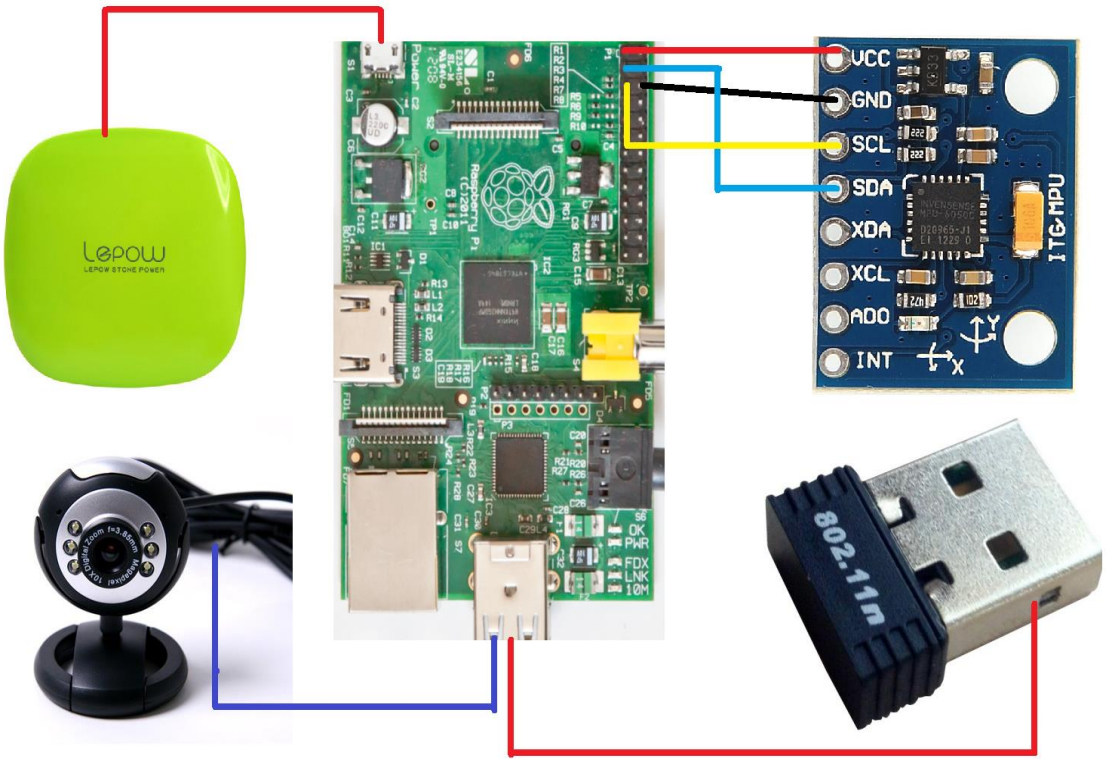

Figure 2 System architecture

The one used in this project is the second version of it. It has an ARM 1176JZF-S processor, which runs at 700MHz clock speed,

a VideoCore IV GPU, 512MB SDRAM shared with GPU, 2 USB port, 1 video and audio output, 1 100 Mbit/s Ethernet port, 1 HDMI output. It also has 26 pins including 8 General purpose Input/output (GPIO), 1 I2C bus, 1 SPI bus, 1 UART bus and 3.3V, 5V and GND. Figure 3 shows the full view of Raspberry Pi. Thus it can be used as a really powerful microcontroller which can fulfill almost any functions, and in the meantime, acting as a normal use computer with keyboard, mouse and monitor connected. Table 1 shows detailed specifications of Pi.

Figure 2: Raspberry Pi components

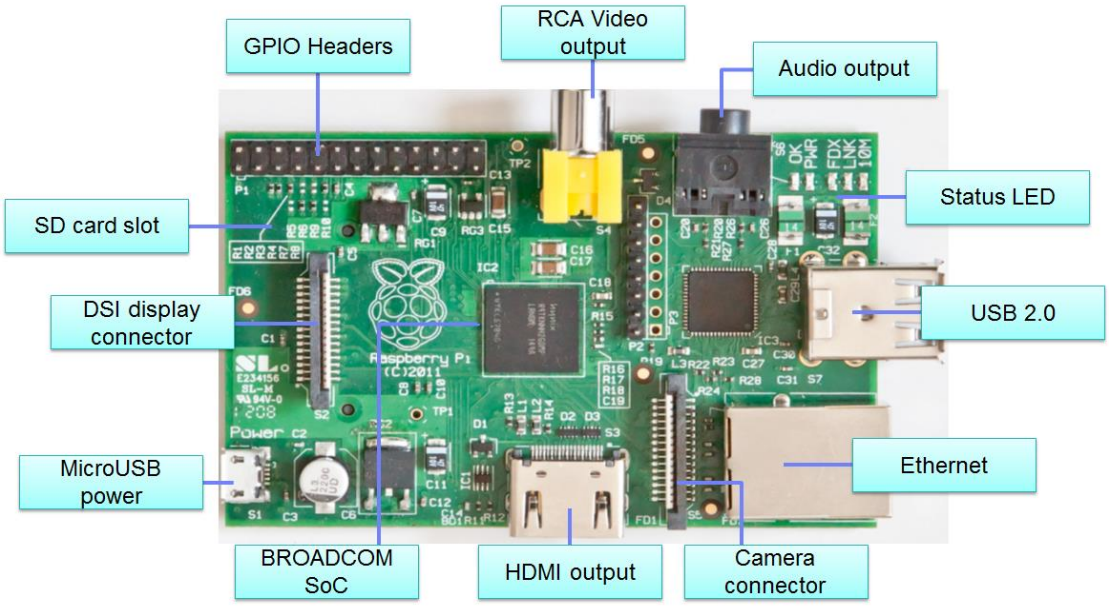

Figure 3: Raspberry Pi components

Raspberry Pi does not have on chip memory, so it needs an external SD card to store either its operating system or all the user data. Raspberry Pi can be used in both ways, either connect to keyboard, mouse and monitor be used as a normal computer or connect it to the

local network so that it can be controlled through SSH terminal. All the specifications would be discussed in the next section.

SoC	Broadcom BCM2835 (CPU, GPU, DSP, SDRAM, and single USB port)
CPU	700 MHz ARM1176JZF-S core
GPU	Broadcom VideoCore IV @ 250 MHz, 1080p30 h.264/MPEG-4 AVC high-profile decoder and encoder[3]
Memory	512MB SDRAM shared with GPU
USB	2
Ethernet	10/100 Mbit/s
Storage	SD card
Low-level peripherals	26 pins including 8 General purpose Input/output (GPIO), 1 I2C bus, 1 SPI bus, 1 UART bus
Power ratings	700 mA (3.5 W)
Power source	5 V via MicroUSB or GPIO header
Size	85.60 mm × 53.98 mm (3.370 in × 2.125 in)
Weight	45 g (1.6 oz)
Operating systems	Arch Linux ARM, Debian GNU/Linux, Gentoo, Fedora, FreeBSD, NetBSD, Plan 9, **Raspbian OS**, RISC OS, Slackware Linux
Price	$35 + $9 WIFI adapter

Table 1: Raspberry Pi specifications

3.2.2 Raspberry Pi using guide

1. Installing an operating system

The system used in this project is Raspbian OS that raspberry foundation supplied. In order to install the operating system, first go to the Raspberry Pi official website and download the latest system image, then use the win32disk imager to write the image into your SD card using your personal computer. Finally if everything goes well, plug the SD card into Pi and the system is good to go. If the status led successfully blink, then the system is installed properly. First time start

up might take a little while, but after that the system will be running really fast.

2. Control Raspberry Pi

In this project, the Pi is used as a totally stand-alone system, so keyboard, mouse and monitor are all not necessary. The Pi will be controlled by using its IP address through the SSH terminal. There are a lot of SSH terminal can be used, Putty is chosen in this project. In addition, eclipse is using to do data transfer between my computer and Pi. This data is the testing programs I wrote in my PC. As I said, the Pi is controlled through its IP address, so internet cable must be plug into the Ethernet port, or a Wi-Fi adapter plug in the USB port. If Pi is in a local network, the IP address can be found by accessing the router, otherwise, a specific software is needed to get the actual IP address of Pi. Figure 4 shows the GUI interface of Putty where you can type in the IP address and access Pi. After access into Pi, user name and password would be needed, in this case, the default user name is pi and the password is raspberry. Figure 5 shows the interface after we access Pi successfully.

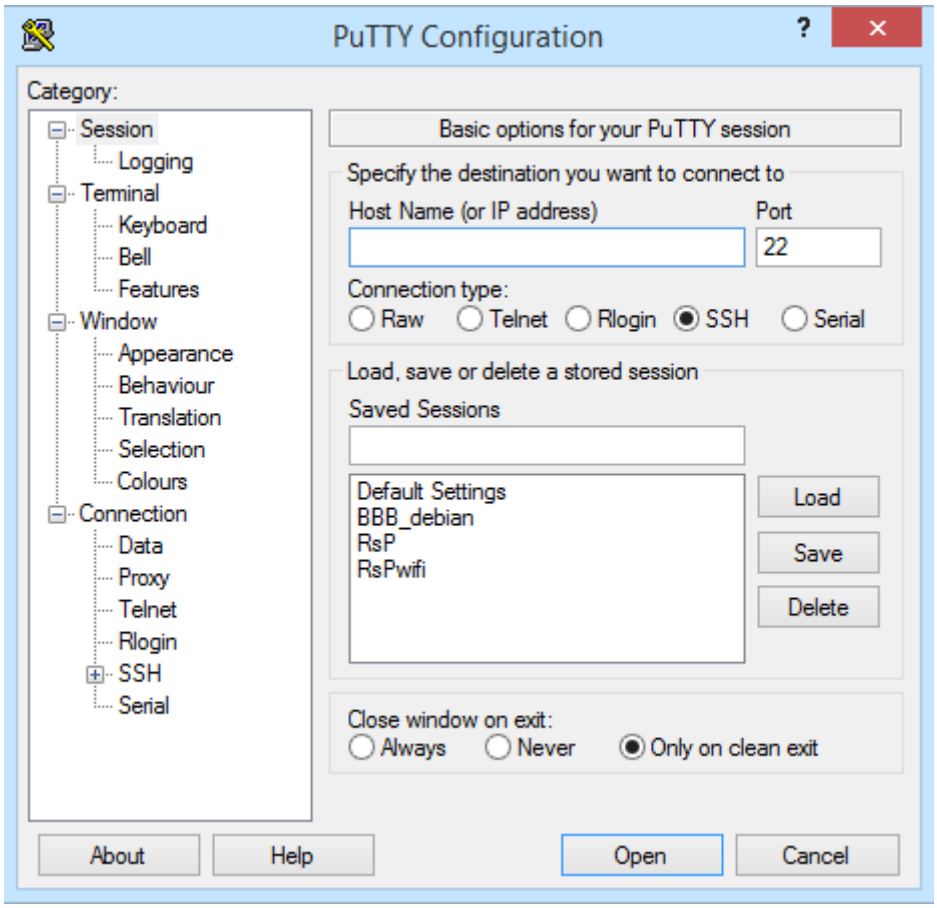

Figure 4: Putty interface

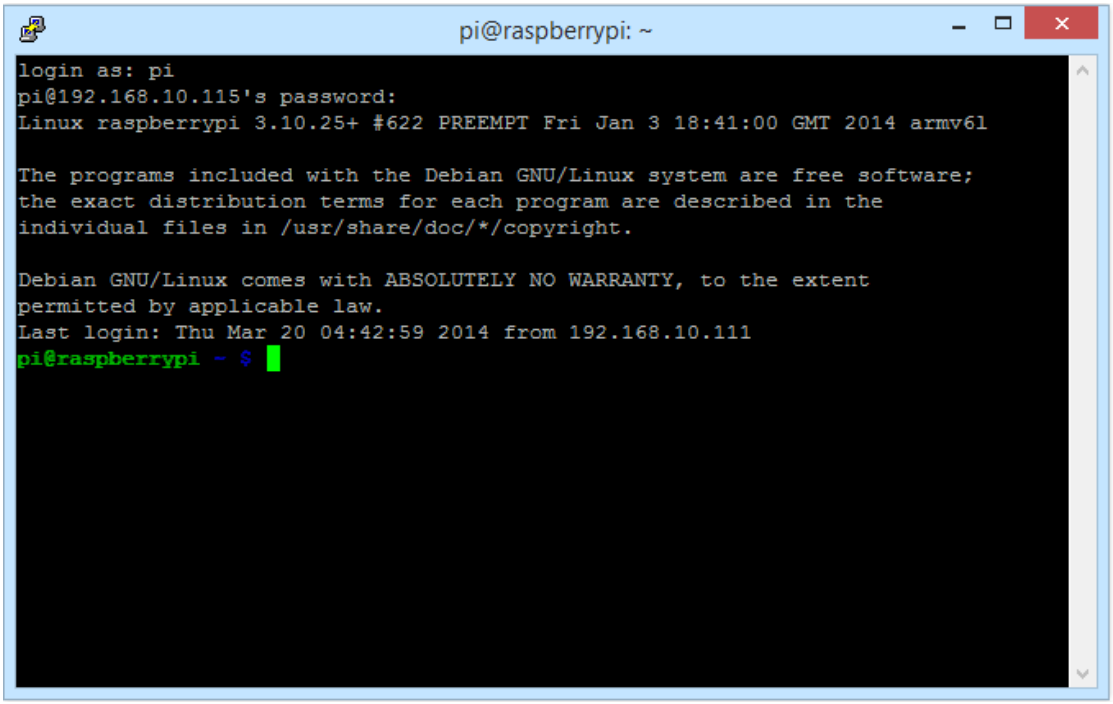

Figure 5: Login session

After accessing into Pi successfully, we can use the Linux system in command line interface. Some basic Linux command will be

introduced in the next session.

3. Linux command introduction

Table 2 shows some basic Linux command used in the project. A lot of different tools are used in this project, including i2c-tools for basic i2c communication, python-smbus for doing i2c in python environment, python-opencv for taking picture using the webcam and ssmtp for sending email using the smtp server. The test program is programmed using python language.

uname -a	Shows you current system version and system time
ls -al	List all your files that are saved in your current directory, including the hidden files, "-a" command is used to list the hidden files and "-l" is used to list the more specific information of each file, including their size, their modified time, etc.
cd	Change the directory to your home directory
cd /bin	Change the directory to /bin
sudo	Often use with other command, means to execute the command as a super user instead of a normal user
rm	Delete the file you don't need any more, followed by the file name in the current directory
sudo apt-cache search i2c	Search all the i2c tools that are available on the system support website
sudo apt-get install i2c-tools	Install the i2c tools we need to communicate with MPU6050
sudo apt-get install python	Install python programming language
python test.py	Execute the test program written in python
sudo apt-get update	Update all the software to the latest version, it is necessary for i2c tools to be used properly
vi	Enter the vim editor to edit any text files or program source code
history	List all the command you executed since the last time you logged in
lsusb	List all the USB devices that are currently plug in the USB port

Table 2: Linux basic command

3.2 MPU6050

MPU6050 is the world's first integrated 6 axis motion sensor, it combines one 3 axis accelerometer and one 3 axis gyroscope, and it has its own digital motion processor (DMP) which can process the motion data with its inside algorithm [2]. It can output 6 axis raw data as well as 6 axis data which pass through the Kalman filter or processed by the Quaternion algorithm. However, access to the filtered data as well as the DMP need specific permission, so only the raw data is used in this project. This sensor can also attach a 3 axis compass through the I2C bus which makes it a 9 axis inertial motion sensor. The chip itself has an internal 16 bit analog to digital converter (ADC), so the output data are 16 bit digital values [2]. There are 117 registers in total inside the chip and all of the registers are 8 bit, so it needs two registers to hold the value for one axis' data. The detection range of the accelerometer is +2g, 4g, 8g,16g and that of the gyroscope is +250, 500, 1000, 2000 ⁰/s, the range can be chosen by setting the corresponding registers. MPU6050 is communicated with Raspberry Pi through the I2C data bus at the clock frequency of 100 kHz. Figure 6 shows the break out board of MPU6050.

The output value of the chip is not the actual acceleration or angular velocity. It has to be calibrated according to the calibration value in the datasheet. For example, if the range of the accelerometer is ±4g (or

±500 for the gyroscope) according to the datasheet, its units is 8192 LSB/g (65.5 LSB/ (⁰/sec) for the gyroscope), which means that if the digital data we get from the chip is 9000, the actual value should be 9000/8192=1.1g for the accelerometer and 9000/65.5=137.4 ⁰/sec for the gyroscope.

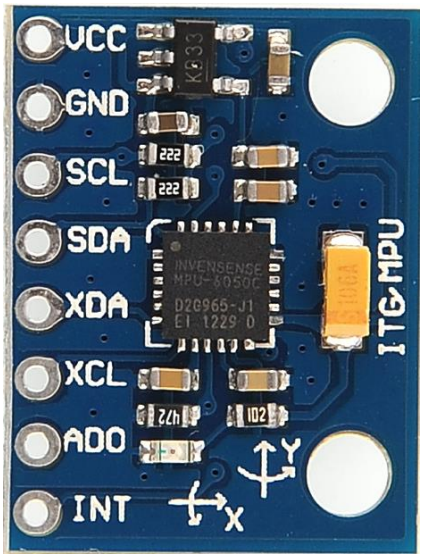

Figure 6 MPU6050 break out board

3.3Webcam

The Webcam used in the device is SANOXY USB 6 LED Night vision PC Webcam.

It is a high performance webcam and PNP(Plug and Play)product for real-time data transmission to PC via USB port. It has a high resolution and fast transmission rate. It uses a color CMOS image sensor with the resolution of 640x480, the video format is 24bit RGB and the videos it make have the specification of 320x240 up to 30frame/sec and 640x480 up to 15 frame/sec(VGA). The S/N ratio is

48dB and the dynamic range is 72dB. Its focus range is 3cm to infinity. It has built-in image compression and automatic white balance. The USB Cable Length is 1.3m. Figure 7 shows the picture of the camera.

Figure 7 USB Webcam

3.4Power supply

The power source used for the device is a 6000mAh external battery for smart phones and tablets. The size of this battery is 3.2 x 3.2 x 0.8 inches and it weighs 9.6 ounces. The input current for this battery is 5V/1000mA. And it has two output ports, one with the current of 5V/1200mA and the other is 5V/2100mA. The 5V/2100mA is just right choice for Raspberry Pi. Figure 8 shows the image of the external power source.

Figure 8 External Battery

4 Results

Lots of Experiments have been done to determine the threshold and the movement trends of different doors. Figure 9, 10 and 11 show the experimental data of moving the door slowly, normally and fast. The experiment condition is in this way, the y axis of the sensor is parallel to the direction of the gravitational force, so the value of that axis is around 16384 (equal to 1g) and that's why I didn't show it in the diagram. Because of that, the door is rotating around the y axis of the sensor. From the figure, we can found out these different characteristics. First, the accelerometer does have some values, and the amplitude of the value is bigger along with the movement, but it does not have a clear pattern like that of the gyroscope, so it is not a good choice for determine the rotational movement. Second, the gyroscope x axis and z axis do not have any value, which makes sense because there is no rotational movement around that two axis. Third, the y axis of the gyroscope has a really clear pattern, it means that in the first two tests, the door is opened and closed for one time, but in the last test, the door is opened and closed twice. The gyroscope can determine the movement pattern really clearly.

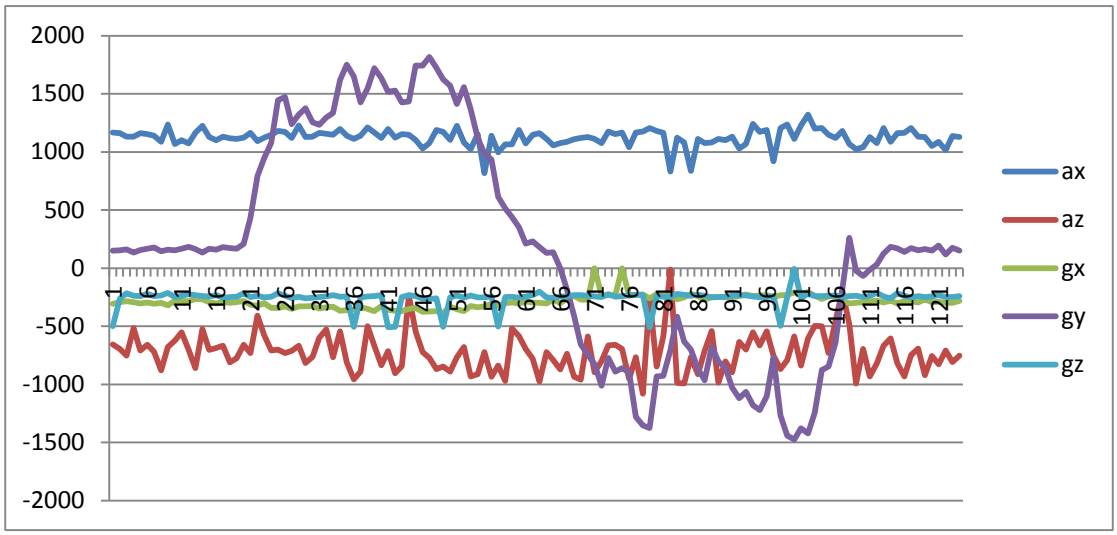

Figure 9 Moving the door slowly

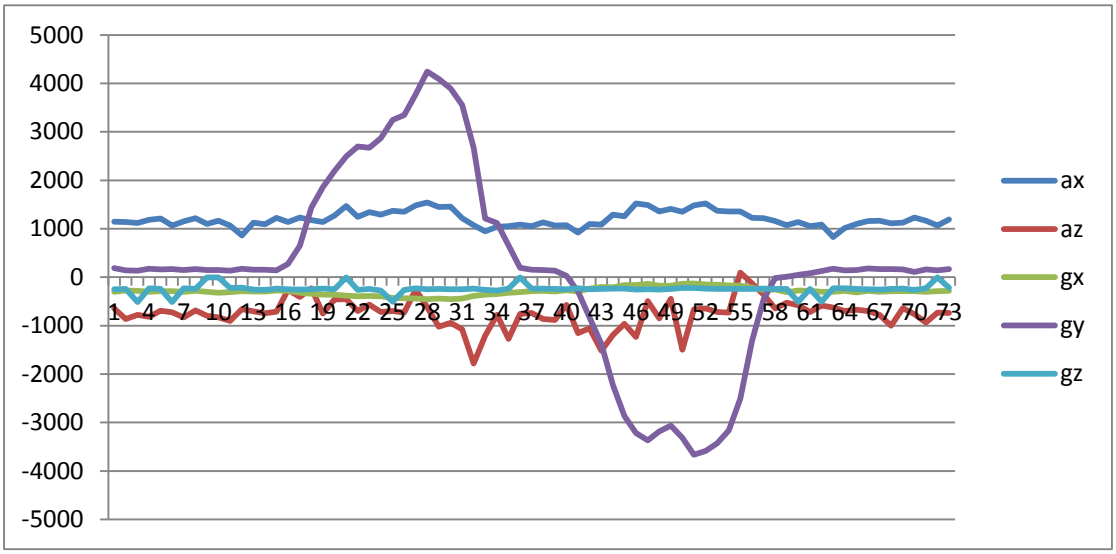

Figure 10 Moving the door normally

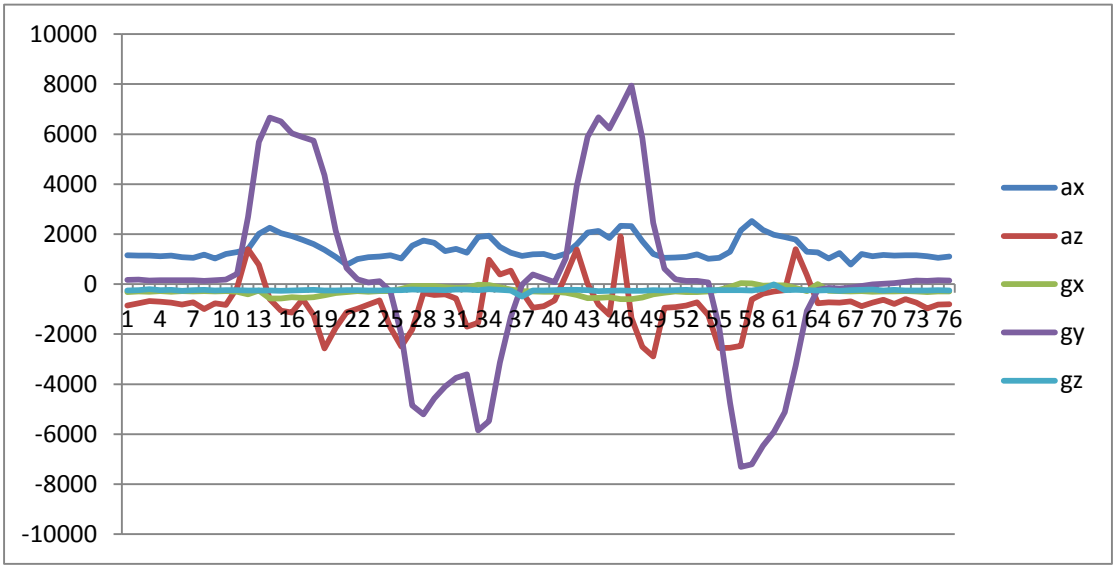

Figure 11 Moving the door fast

Forth, the amplitude of the y axis gyroscope value increases as the

door movement speed increase. Thus there are two ways to detect the door movement, wither by monitoring the whole pattern of the door movement or by set a single threshold value. The second way is chosen in this project and the threshold value is 1500. It is not too high to miss the slow motion values but also not too low to detect unexpected motions. Furthermore, the y axis of MPU6050 is parallel to the direction of the gravitation in the experiment set, however in actual use, the device can be placed in any direction, thus the gyroscope data has to be normalized by equation 1. By implementing the whole device on the door or cabinet door, it can detect the motions and send out the alert email. Figure 12 shows the final setup of the device.

$$g\_normal = \sqrt{g_x{}^2 + g_y{}^2 + g_z{}^2} \quad (1)$$

Figure 12 System setup

5 Summary of portable door monitoring device for home and office security

This system is a totally portable device, it has small size and can be placed on any kind of doors and detect the motion successfully. Because of the feature that the webcam only works when signal is detected, it can save a lot of power compare to that of the normal surveillance system. The device is really cheap for residential use, the total system setup will need 50 US dollars for now, but when it is in commercial use and has only one break out board for the whole system, it will be much cheaper and much smaller. This device can also send out the alert email when a break in occurs immediately with about 10 seconds of the email delay. And if the camera is placed properly, it can take a picture or video of the theft's face and is much more effective than that of the surveillance system.

Part 2 Portable 3D mouse

1 Introduction

Mouse has been invented as a way to control the PC since 1968, from the mechanical types to nowadays optical mouse, it becomes more and more accurate and brings people a lot of convenience. Though it is a really good way to control the computer, it still has some shortcomings, for example, it always needs a surface to work on, or it may cause the so called “computer mouse fatigue” after a long period of using for our wrist or arm [18]. So it is quite essential to develop a mouse that can be easier to use and does not have too many limits (work surface, etc). Then the 3D mouse comes into my thought, which can just be used in the air to control the PC for normal use. The idea of this mouse is to use some inertial sensors to model the finger move so that it can control the mouse curser, just like our finger is moving around the screen which is apparently more convenient compared to the old style surface mouse [28]. MPU6050, which is an inertial motion unit (IMU) contains an accelerometer and gyroscope, is used to sense and model the finger move; and MSP430G2 is used as the microcontroller to control the MPU6050, do the data processing and communicate with PC through the COM serial port, a smoother filter

and a complementary filter (data fusion) are used in MSP430 to make the mouse more accurate.

2 Methodology

For a system that can be used as mouse and does not need a working surface, the hand movement detection is quite necessary. Thus MPU6050 is used as the sensor to detect every single move of the hand in the air. In this case, for the purpose of accurately detection, both of the accelerometer and gyroscope are used. And their raw data is not accurate enough. So one smooth filter is used to smooth the data and one complementary filter is used to do the data fusion for two types of the data [25]. After filtering the data, MSP430 would send the filtered data to PC COM port through the UART (Universal asynchronous receiver/transmitter) bus. A PC COM driver is written in C++ MFC to read the serial value and to be applied to the mouse move equation. For the equation, accelerometer data is used as the acceleration and gyroscope data is used as the velocity. The actual data of the gyroscope is angular velocity, so basically it cannot be used as velocity to compute the displacement of our hand, but it is true that when the sensor is used as a mouse, it is moved in really small angle, thus in that range, , $\theta = \sin\theta$ so the angular velocity can just be used as the moving velocity of our hand.

The 3D mouse system is implemented by MSP430G2553 launchpad and the MPU6050 IMU (inertial motion unit) module. The two devices are connected through the I2C communication protocol, more

detail will be provided in the following section about I2C. The whole system works in this way. The MPU6050 would sample the accelerometer and gyroscope at a rate of 1 kHz and then convert the analog signal through its internal 16 bit ADC, so for each axis of the accelerometer and gyroscope data, the output should be 16 bit digit signal. This digital signal will be received and processed by MSP430 through the I2C communication at a rate of 100 kHz. After the MSP430 received the IMU data, it will transfer the data to the computer com port through its hardware UART (Universal Asynchronous receiver/transceiver) port. More detail will be provided in the following section about UART. PC would process the motion data through our algorithm (Programmed in C++) to control the mouse.

3 System Architecture

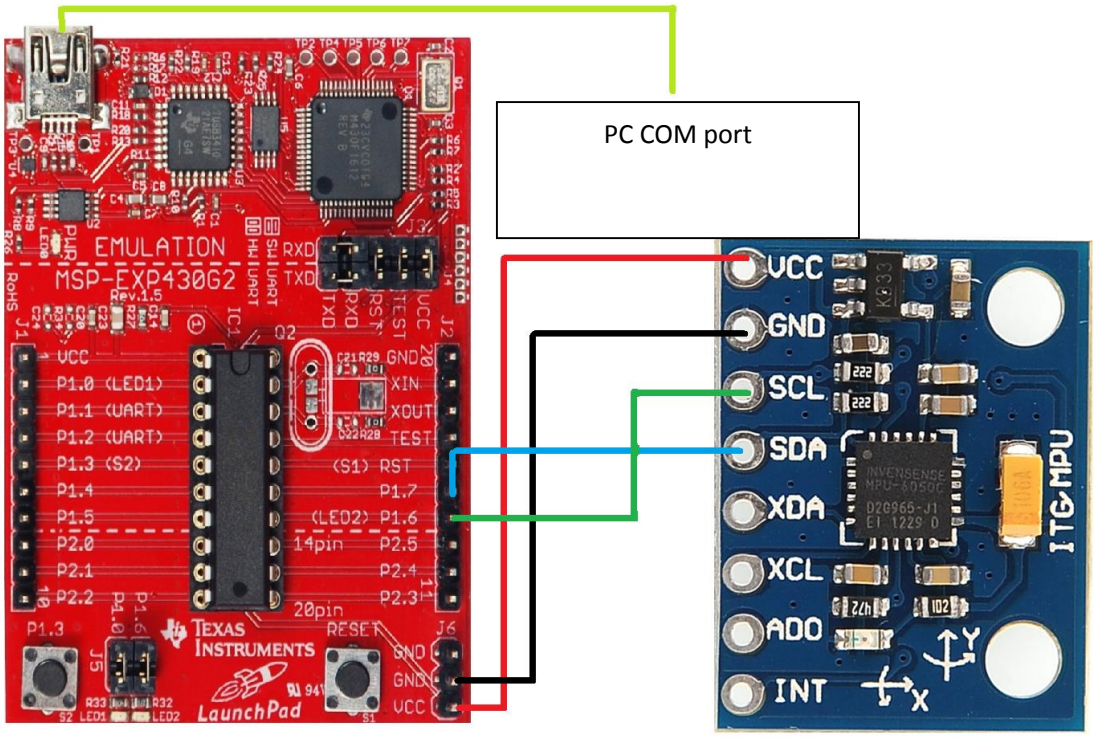

Figure 13 3D mouse system architecture

The whole system is composed of two parts, the MPU6050 to sense the hand movement and it is connected to MSP430 through I2C bus. The MSP430G2553 do the data processing and pass the data through UART bus to PC COM port. Figure 13 shows the image of the whole system. The blue one is the MPU6050 and the read one is the MSP430 with G2553 chip on board.

3.1MSP430G2553 Introduction

The MSP430G2553 is a 16-bit RISC (Reduced instruction set) CPU [1], Figure 14 shows the architecture of MSP430G2553. From the figure we can find that it has its own JTAG debugger for the ease of hardware debug, it has a really flexible clock system which contains 3

clock sources, the ACLK (auxiliary clock) coming from the external 32kHz watch crystal, and the MCLK (main clock) for hardware use and the SMCLK (sub-main) clock for software use that both come out of a digitally controlled oscillator. These three clocks can all be divided by 1, 2, 4, or 8. However, only the ACLK and SMCLK can be used as the software selectable clock source. It also has an on chip flash of size 16KB which is enough to fit the code in. Furthermore, the RAM size is 512 byte, it is really small that even a printf function cannot be used in the software. The Launchpad also has a watch dog timer that can be used as a watch dog to detect software errors or used as a normal interval timer [1]. The other peripherals are the GPIO (General purpose Input/Output) headers that includes one I2C bus, one SPI bus and one UART bus. Figure 15 is the pin diagram of MSP430. From the figure it is obvious that this chip has 8 channel of ADC(Analog to Digital Converter), the resolution of this ADC 10 bit, and it can work in four different modes. Thus this device is a really good choice of mixed signal application. The G2553 chip has four different low power modes and one active mode. When it is in active mode, the current is only around 300microA at voltage of 3V. Therefore it is a well-designed lower power consumption device.

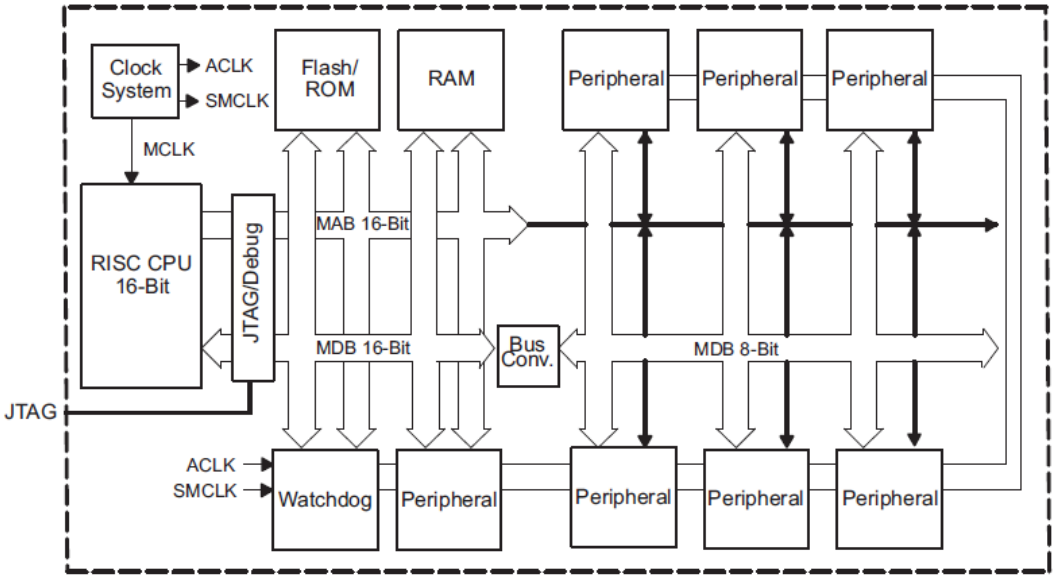

Figure 14 MSP430 Architecture

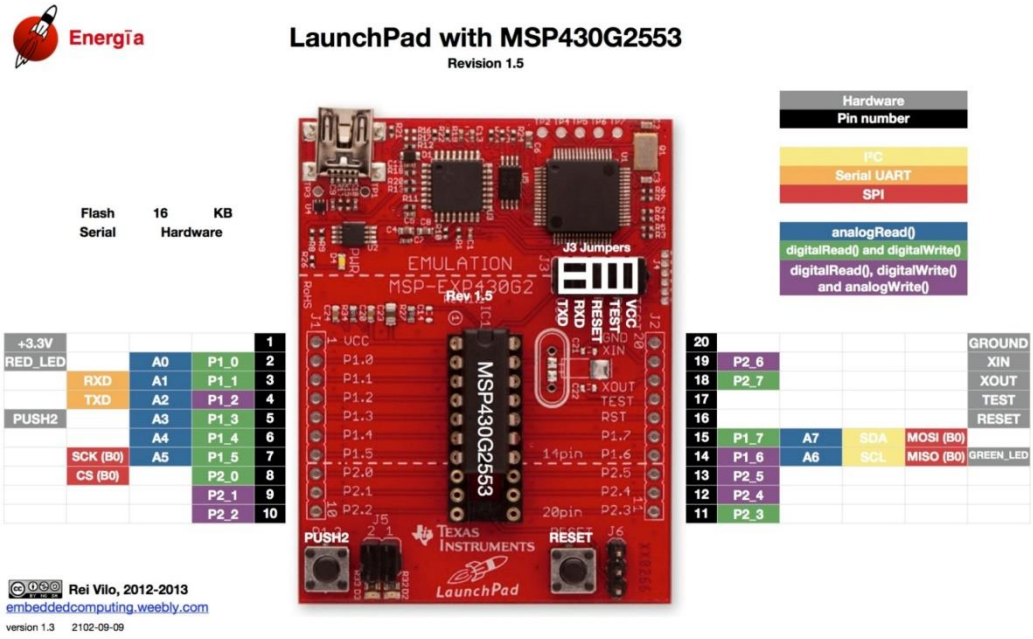

Figure 15 MSP430 Pin Diagram

3.2 I2C Introduction

In I2C mode, the USCI (Universal Serial Communication Interface) module provides an interface between the MSP430 and

I2C-compatible devices connected by way of the two-wire I2C serial bus [1]. External components attached to the I2C bus serially transmit and/or receive serial data to/from the USCI module through the 2-wire I2C interface.

7 bit or 10 bit address modes are both supported in the I2C module, it also includes the START/ RESTART/ STOP and the Multi-master transmitter/ receiver mode. What will be used in the 3D mouse is 7 bit address mode and MSP430 would be acting as the master which will control the clock source and indicates the data transfer. As what shows in the figure 16, I2C data is communicated using the serial data pin (SDA) and the serial clock pin (SCL). Both SDA and SCL are bidirectional, and must be connected to a positive supply voltage using a pull-up resistor.

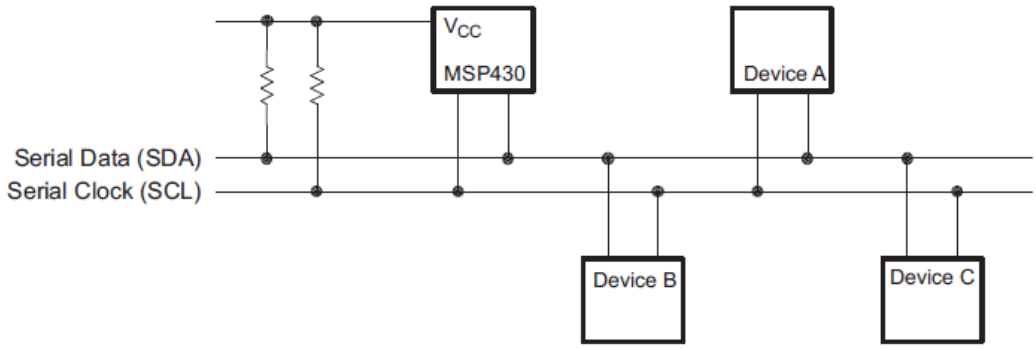

Figure 16 I2C connection diagram

The I2C mode operates with byte data. Most significant bit is transferred first. For each data bit transferred, one clock pulse is generated by the master device. The first byte after a START condition

(generated by the master, when SCL is high, SDA turns from high to low) consists of the R/W bit and a 7-bit slave address. When R/W = 0, the master transmits data to a slave. When R/W = 1, the master receives data from a slave. The master will receive an ACK bit after each byte on the 9th SCL clock, thus for 1 byte data transfer, there is actually 9 bit in total. As what's shown in the following figure.

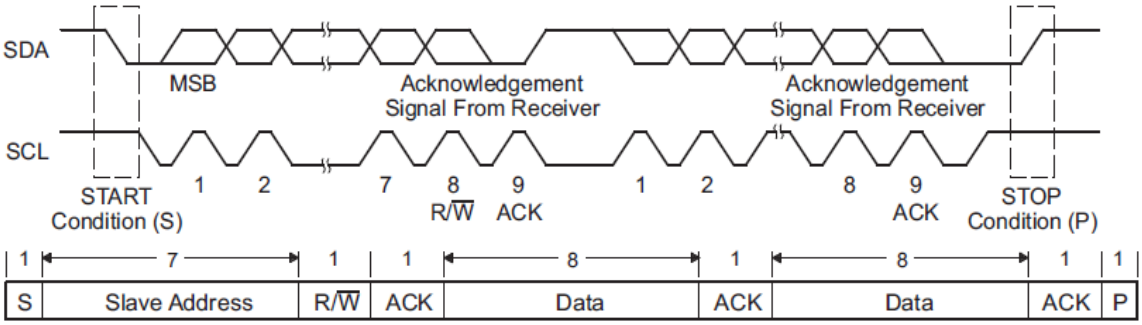

Figure 17 I2C data transmission

Data on SDA must be stable during the high period of SCL as shown in Figure 17 [1]. The SDA signal can only change when SCL is stable at low, otherwise it will generate the START or STOP conditions.

The master can issue a restart signal so that without first stopping a transfer, it can change the direction of SDA data flow for either write or read, the restart signal contains the slave address of the new data register as well as the direction specified by the R/W bit.

The initialization of I2C in MSP430G2 including the following steps [1]:

- Set UCSWRST, Initialize all USCI registers with UCSWRST=1 (including UCB0CTL1)

- Set UCB0CTL0 register as master synchronous mode
- Set UCB0CTL1 as SMCLK
- Set UCB0BR0 to set the clock frequency (10 for 100kHz)
- Configure ports. (BIT6 + BIT7)
- Set the UCB0I2CSA register as the slave address
- Clear UCSWRST via software (UCxCTL1 &=~ UCSWRST)
- Enable interrupts (optional) via UCxRXIE and/or UCxTXIE

The code would be included in the attachment section.

3.2.1 Master Transmitter and Receiver

MSP430 can work both as a master transmitter and receiver. After initialization, master transmitter or receiver mode is initiated by setting the UCTR register. MSP430's USCI module will check whether the bus is available or not, if it is, the START condition will be generated by setting the UCTXSTT register. The data received from the slave device is stored in the UCBxRXBUF and user can read it in the C program, the data that MSP430 want to send to the slave device is written in the UCBxTXBUF. There are also flag registers for both of the buffer to indicate whether the buffer is empty or not so that user can check the data efficiently.

The USCI module checks if the bus is available, generates the START condition, and transmits the slave address. The UCBxTXIFG bit is set when the START condition is generated and the first data to be transmitted can be written into UCBxTXBUF. As soon as the slave

acknowledges the address the UCTXSTT bit is cleared. After data transfer finished, s STOP condition can be generated by setting the UCTXSTP register. One important thing to point out is that if a master wants to receive a single byte only, the UCTXSTP bit must be set while the byte is being received. For this case, the UCTXSTT may be polled to determine when it is cleared [1].

3.3 UART introduction

In asynchronous mode, URXD and UTXD are the two external pins for MSP430 to connect with an external system [1], the UART transmits and receives characters at a bit rate asynchronous to the other device.

The MSP430G2 UART includes 7- or 8-bit data with odd parity, even parity, or non-parity, independent transmit and receive shift registers and buffer registers. It also supports programmable baud rate with modulation for fractional baud rate support and independent interrupt capability for receive and transmit.

The UART character format, shown as in the following figure 18, consists of a start bit, seven or eight data bits, an even/odd/no parity bit, an address bit (address-bit mode), and one or two stop bits. The UCMSB bit controls the direction of the transfer and selects LSB or MSB first. LSB-first is typically required for UART communication.

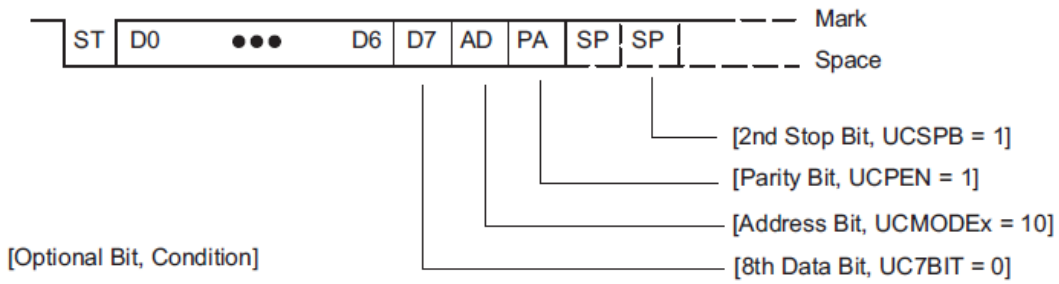

Figure 18 UART data specification

The initialization of UART in MSP430G2 including the following steps:

- Set UCSWRST, Initialize all USCI registers with UCSWRST=1.
- Set UCA0CTL1 register to use the SMCLK.
- Set UCA0BR0 and UCA0BR1 to 104 to set the baud rate as 9600.
- Configure ports (BIT1 + BIT2).
- Clear UCSWRST via software (UCxCTL1 &=~ UCSWRST).
- Enable interrupts (optional) via UCAxRXIE and/or UCAxTXIE.

By clearing the UCSWRST can enable the USCI module. The transmitter and receiver are ready and in an idle state. The transmit baud rate generator is ready as well for the UART mode.

There’s no start condition for UART like that of the I2C bus, by writing data to the UCAxTXBUF register, a transmission can be started, and in the meantime, the baud rate generator is enabled. Then, data in UCAxTXBUF will be moved to the transmit shift register on the next clock. As long as there is new data in the transmit buffer, the transmission will be continued. The transmitter will return to its idle

state when the transmit buffer is empty.

The UCAxRXBUF register is used to store the value it receives from external devices. Every time a data is received, the UCAxRXIFG register will be set to indicate that the data is ready to be read.

MPU6050 Introduction

See section 3.2 in the part 1.

3 Results

4.1 Smooth filter

The accelerometer can detect acceleration really accurately, but it suffers the vibration error. It means that even the accelerometer is placed stable on the table, there would be still some high frequency values. Especially when human hand is moving the accelerometer, the data would be quite unstable, so the smooth filter is applied to the accelerometer data. It is filtered by the equation 2.

$$\begin{aligned} &\text{Accel\_pre} = \text{Accel\_filtered} \\ &\text{Accel\_filtered} = \text{Accel\_pre} + 0.6 \cdot (\text{Accel\_raw} - \text{Accel\_pre}) \end{aligned} \tag{2}$$

Figure 19 shows the filtered result. From the diagram we can find that the filtered result is much better than the original one.

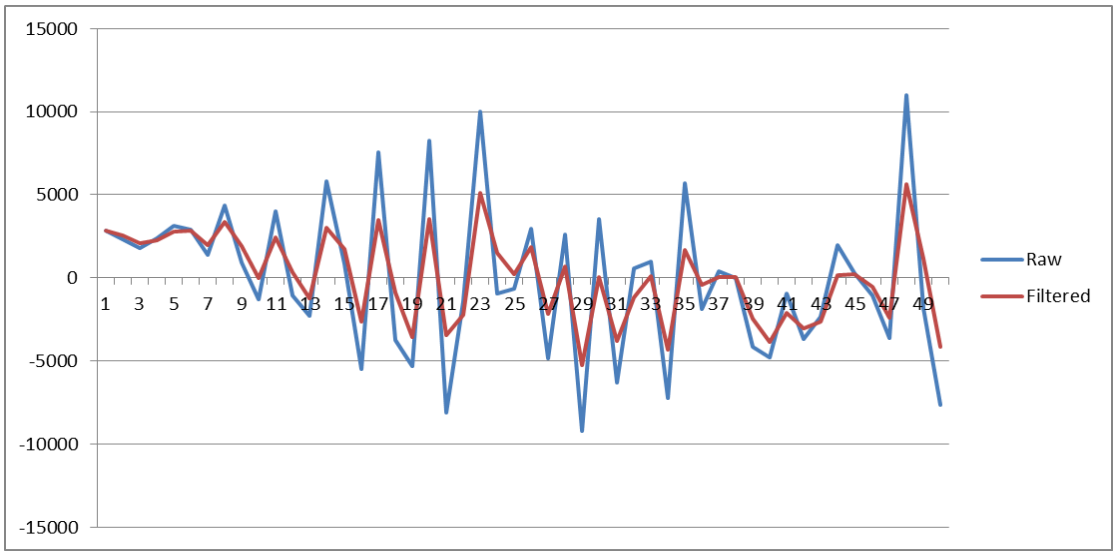

Figure 19 Smooth filter result

4.2 Complementary filter

The current algorithm only use the data from some specific axis, this need the user hold the MPU6050 in specific way, but normally when people are using the mouse, they want to hold in the way they like. Therefore we have to know the position angle of MPU6050 and then normalize the 6 axis data so that no matter how the device is holding, it will always work well. In order to detect the angle of MPU6050, the accelerometer can do it directly by determine the position of gravity vector which is always visible for the accelerometer, but as previously stated, the accelerometer suffers any kind of vibrations and always has a high frequency noise, so it is good for long time use but not for short time. The gyroscope can also determine the angle by integrate its value with time. The problem of gyroscope is that it has a drift after certain period of use because of the integrating error. So the gyroscope is good for short term use but not for long time. Figure 20 and 21 shows the accelerometer vibration noise and the gyroscope drift problem when calculating angles.

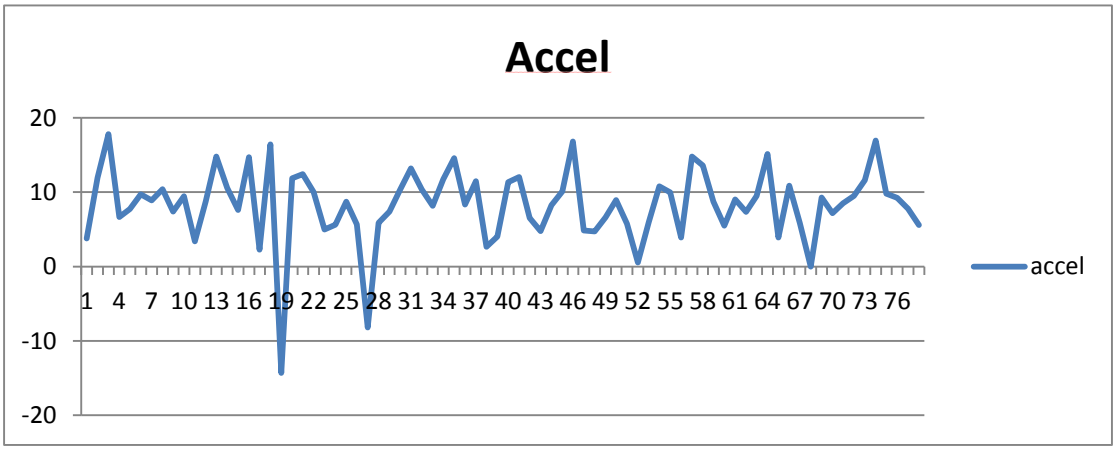

Figure 20 Accelerometer noise

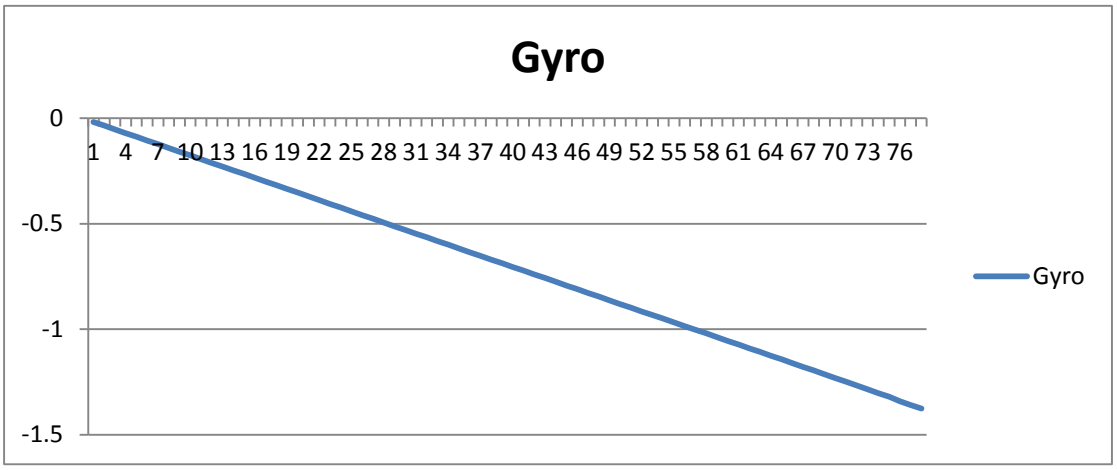

Figure 21 Gyroscope drift

Thus the two data can be combined with each other and complement each other's weak points. That is the complementary filter. Figure 22 shows the diagram of the complementary filter. The accelerometer pass through a low pass filter and the gyroscope pass through a high pass filter. The two combined together can determine the angle really accurately.

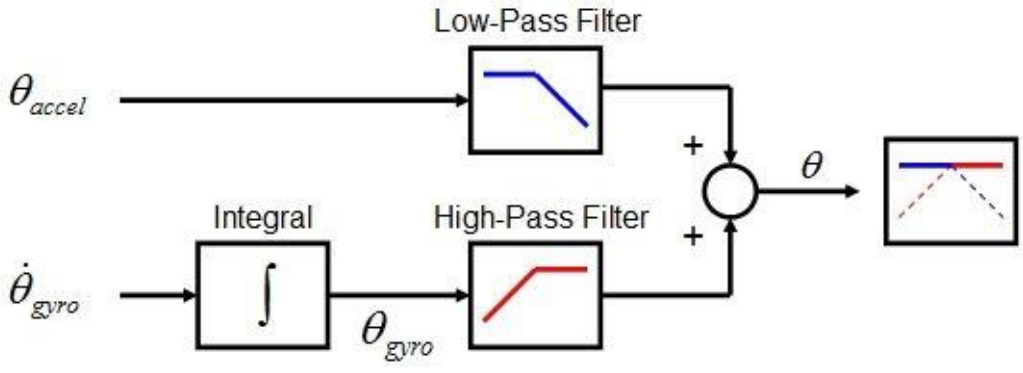

Figure 22 Complementary filter

Equation 3 show how to implement the complementary filter. The gyroData is the values from the MPU6050, the angle is the angle that

is calculated last time and dt is the loop constant, normally is 0.01s.

$$angle = 0.98 \bullet (angle + \text{gyroData} \bullet \text{dt}) + 0.02 \bullet (\text{accData}) \quad (3)$$

The accData is the angle that is calculated by the accelerometer data. As shows in Figure 23, the angle with x, y and z axis of MPU6050 is ρ, φ and θ, respectively. They are calculated by the following equations.

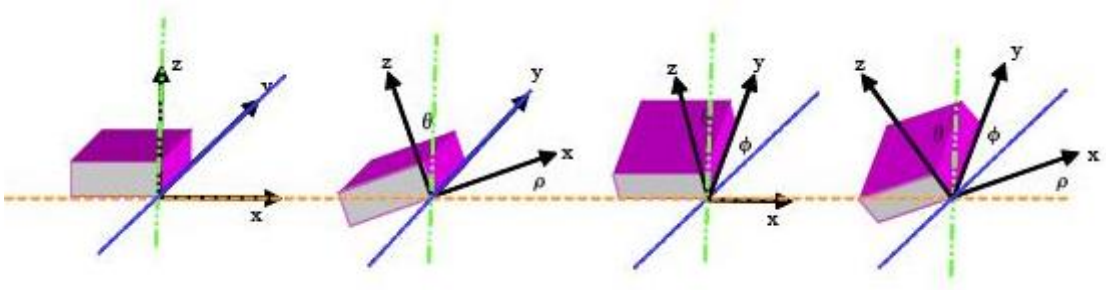

Figure 23 MPU6050 angle

$$\left.\begin{cases} a_x^{\,2} + a_y^{\,2} + a_z^{\,2} = g^2 \\ a_x = g \bullet \sin\rho \end{cases}\right\} \rightarrow a_y^{\,2} + a_z^{\,2} = g^2 \bullet \cos^2\rho = \frac{g^2 \bullet \sin^2\rho}{\tan^2\rho} \quad (4)$$

$$\rho = \arctan\left(\frac{a_x}{\sqrt{a_y^{\,2} + a_z^{\,2}}}\right) \quad \phi = \arctan\left(\frac{a_y}{\sqrt{a_x^{\,2} + a_z^{\,2}}}\right) \quad (5)$$

Figure 22 shows the result of the complementary. From the figure it can be found that the angle calculated by accelerometer is accurate but not stable, the angle calculated by the gyroscope is drift away from the real angle and the angle after the complementary is almost the same as that of the accelerometer one but is much smoother and stable than the accelerometer data. It is quite obvious that the complementary filter works great. The data in figure 24 is only for a short time, will the data still drift away after a long period of use? Actually not, Figure 25

shows the diagram of long time use. From the figure it is clearly that

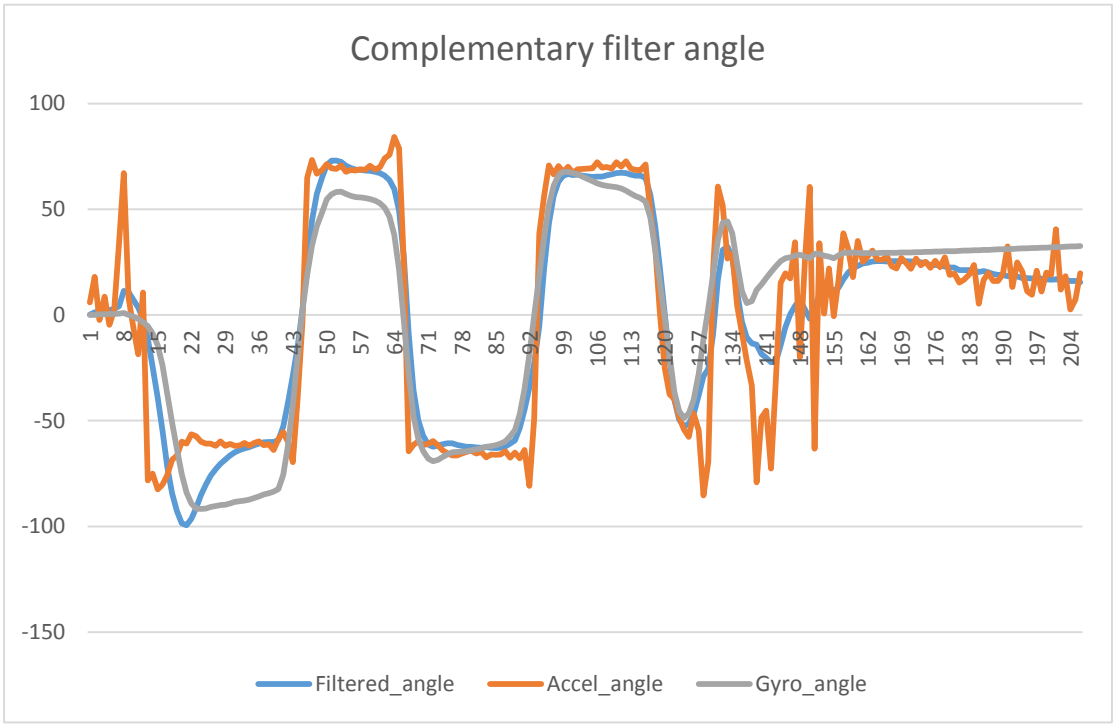

Figure 24 Angle calculated by different ways

the data will not drift even after almost 7 minutes of use.

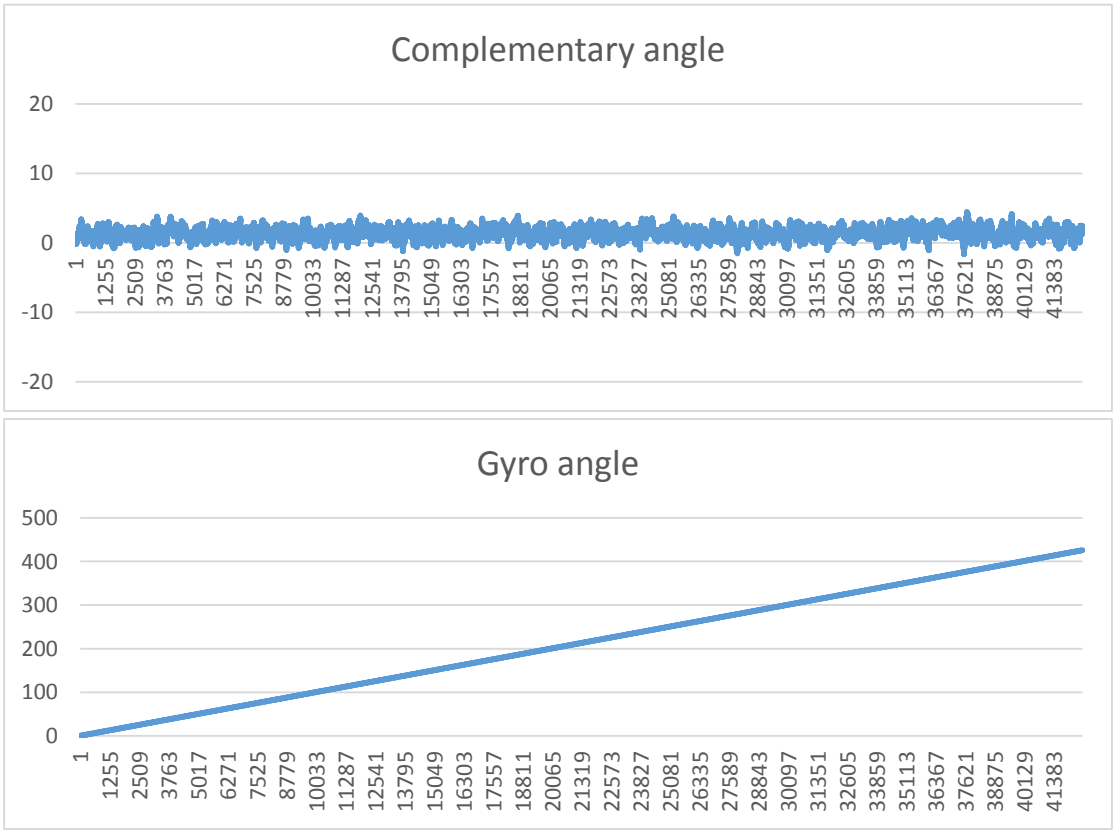

Figure 25 Drift problem

In the actual use, the high pass filter parameter is chosen as 0.98. What is the influence of this parameter on the final result? Figure 26 shows the diagram of different parameters.

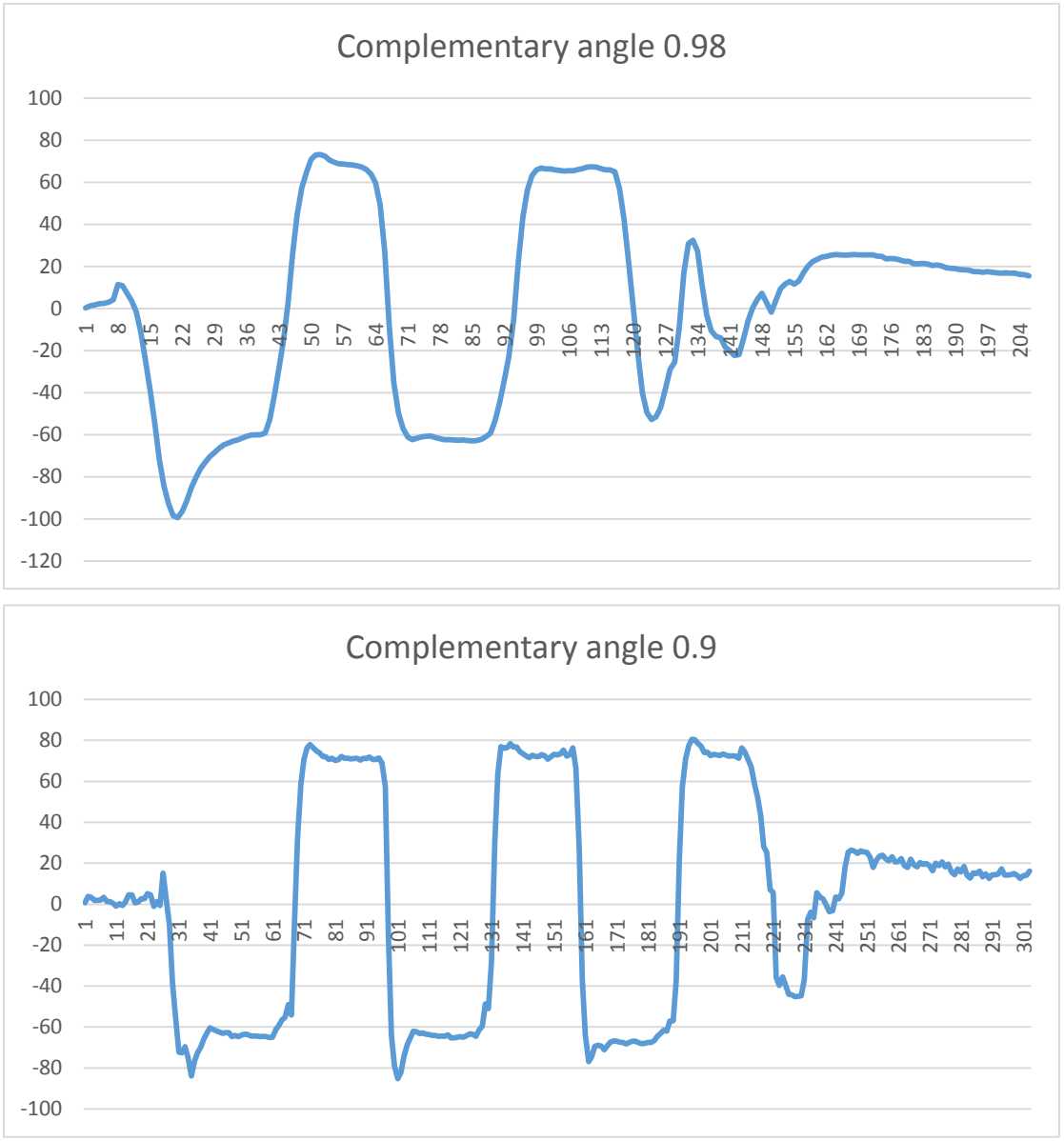

Figure 26 Filter parameter chosen

This figure shows that when the parameter is chosen at 0.9, the angle data starts to be unstable. It means that as the high pass filter parameter gets low, the low pass filter parameter get high and more

accelerometer data are passed through the low pass filter, thus the final complementary angle relies more on the accelerometer, and that's why when the high pass filter parameter gets low, the final angle starts to have some noise. In addition, after several experiments, the parameter of 0.98 is chosen as the optimal value.

After the hardware is ready for use, the 3D mouse algorithm should be implemented to fulfill the mouse control task. A simple C++ com port program is used to obtain the 6 axis filtered motion data continuously can apply the data to a mouse curser movement equation. The displacement of MPU6050 is calculated by the following equation:

$$S = v \bullet t + \frac{1}{2} a \bullet t^2 = g \bullet A + a \bullet B \tag{6}$$

Here A and B are just normalization constant and g is the gyroscope data used as the velocity and a is the accelerometer data used as the acceleration.

Figure 27 is the whole system setup.

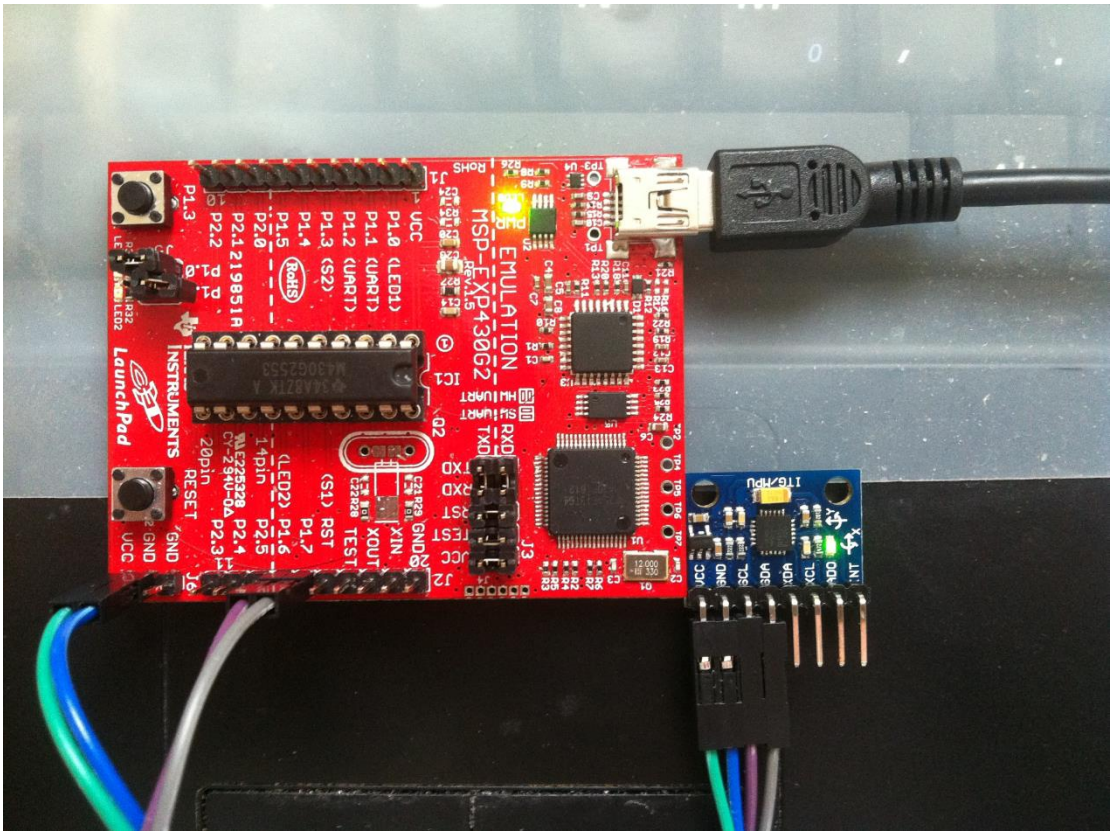

Figure 27 3D Mouse System Setup

5 Summary of portable 3D mouse

The system is a portable mouse that has an acceptable accuracy for normal use. It does not need any work surface and is great for tight work space use as well as the place such as airplane, sofa, etc. Smooth filter and complementary filter are applied in the system and the accuracy is improved compared to that without the filter. The system for now is not totally portable, but is easy to be extended to a wireless mouse with extremely small size that can be wearable, as well as a energy efficient device for long duration use.

6 Conclusion

Two portable home use smart electronics are designed and implemented. The portable monitoring system for home and office security is a totally portable device, it has small size and can be placed on any kind of doors and detect the motion successfully. Because of the feature that the webcam only works when signal is detected, it can save a lot of power compare to that of the normal surveillance system. The device is really cheap for residential use, the total system setup will need 50 US dollars for now, but when it is in commercial use and has only one break out board for the whole system, it will be much cheaper and much smaller. This system can also send out the alert email when a break in occurs immediately with about 10 seconds of the email delay. And if the camera is placed properly, it can take a picture or video of the theft's face and is much more effective than that of the surveillance system.

The 3D mouse is successfully implemented and the two different filters of smooth filter and complementary filter are implemented and make the device accurate enough. This mouse is a small portable device that does not need a working surface and can be extended to a portable mouse that is as small as a ring.

Reference

[1] MSP430G2 family User Guide.

[2] MPU6050 Data Sheet and Register Map.

[3] F. Zuo and P. H. N. de With, “Real-time Embedded Face Recognition for Smart Home”, IEEE Transactions on Consumer Electronics, Vol. 51, No. 1, pp. 183–190, February 2005.

[4] J. Hou, C. Wu and Z. Yuan, “Research of Intelligent Home Security Surveillance System Based on ZigBee”, In Proceedings of Intelligent Information Technology Application Workshops, pp. 554–557, December 2008.

[5] W. T. Higgins. “A Comparison of Complementary and Kalman Filtering”, In Proceedings of IEEE Transactions on Aerospace and Electronic Systems, Vol. 11, Issue 3, pp. 321–325, May 1975.

[6] A. Benini , A. Mancini , A. Marinelli , S. Longhi, “A Biased Extended Kalman Filter for Indoor Localization of a Mobile Agent using Low-Cost IMU and UWB Wireless Sensor Network”, Robot Control, Vol. 10, Part 1, pp. 735–740, September 2012.

[7] W. Chen, P. Chen, W. Lee, C. Huang, “Design and Implementation of a Real Time Video Surveillance System with Wireless Sensor Networks”, In Proceedings of IEEE Vehicular Technology Conference, pp. 218–222, May 2008.

[8] A. Hampapur, L. Brown, J. Connell, A. Ekin, N. Haas, M. Lu, H.

Merkl and S. Pankanti, “Smart Video Surveillance: Exploring the Concept of Multiscale Spatiotemporal Tracking”, IEEE Signal Processing Magazine, Vol. 22 , Issue 2, pp. 38–51, March 2005.

[9] F. Mei, X. Shen, H. Chen, Y. Lu, “Embedded Remote Video Surveillance System Based on ARM”, Journal of Control Engineering and Applied Informatics, Vol. 13, No.3, pp. 51-57, May 2011.

[10] M. Kumar, N. Murthi Sarma, Ch. Sridevi, A. Pravin, “ARM9 Based Real Time Embedded Network Video Capture and SMS Alerting System”, International Journal of Research in Computer and Communication Technology, Vol. 1, Issue 7, pp. 489–493, December 2012.

[11] J. Zhao, S. Sun, Y. Feng, R. Luan, W. Zhang, “The Design and Realization of Embedded Wireless Video monitoring System Based on GPRS”, In Proceedings of IEEE Wireless Communications, Networking and Mobile Computing Conference, pp. 1-4, October 2008.

[12] H. Huang, S. Xiao, X. Meng, Y. Xiong, “A Remote Home Security System Based on Wireless Sensor Network and GSM Technology”, In Proceedings of IEEE Networks Security Wireless Communications and Trusted Computing Conference, Vol. 1, pp. 535-538, April 2010.

[13] D. Zhou, G. Tan, “Network Video Capture and Short Message Service Alarm System Design Based on Embedded Linux”, In Proceedings of IEEE Natural Computation International Conference, Vol. 7, pp. 3605-3608, August 2010.

[14] S. K. Reddy, “FPGA and GSM Implementation of Advanced Home Security System”, International Journal of Research in Engineering and Applied Sciences, Vol. 3, Issue 1, January 2013.

[15] S. Kawakamia, K. Okanea, T. Ohtsuki, “Detection Performance of Security System Using Radio Waves Based on Space-time Signal Processing”, In Proceedings of International Conference on Security Camera Network, Privacy Protection and Community Safety, Vol. 2, Issue 1, pp. 171-178, December 2009.

[16] M. H. Assaf, R. Mootoo, S. R. Das, E. M. Petriu, “Sensor Based Home Automation and Security System”, In Proceedings of IEEE Instrumentation and Measurement Technology Conference, pp. 722-727, May 2012.

[17] N. Gilbert “Computer Mouse Fatigue”, Desktop and Portable Computers, August 2011.

[18] P. W. Johnson, S. L. Lehman, D. M. Rempel, “Measuring Muscle Fatigue During Computer Mouse Use”, In Proceedings of IEEE Engineering in Medicine and Biology Society, Bridging Disciplines for Biomedicine International Conference, Vol. 4, pp. 1454–1455, November 1996.

[19] P. Lin, H. Komsuoglu, D. E. Koditschek, “Sensor Data Fusion for Body State Estimation in a Hexapod Robot With Dynamical Gaits”,

Robotics, IEEE Transactions, Vol. 22, Issue 5, pp. 932-943, October 2006.

[20] D. Jurman, M. Jankovec, R. Kamnik, M. Topič, “Calibration and Data Fusion Solution For the Miniature Attitude and Heading Reference System”, Sensors and Actuators A: Physical, Vol. 138, Issue 2, pp. 411–420, August 2007

[21] E. Foxlin, “Intertial Head-Tracker Sensor Fusion by a Complementary Separate-Bias Kalman Filter”, In Proceedings of IEEE Virtual Reality Annual International Symposium, pp. 185-194, April 1996.

[22] I. Zunaidi, N. Kato, Y. Nomura, H. Matsui, “Positioning System for 4-Wheel Mobile Robot: Encoder, Gyro and Accelerometer Data Fusion With Error Model Method”, Chiang Mai Univ. Journal, Vol. 5, No. 1, pp. 1-14, June 2006.

[23] F. Caron, E. Duflos, D. Pomorski, P. Vanheeghe, “GPS/IMU Data Fusion Using Multisensor Kalman Fltering: Introduction of Contextual Aspects”, Information Fusion, Vol. 7, Issue 2, pp. 221-230, June 2006.

[24] S. You, U. Neumann, “Fusion of Vision and Gyro Tracking for Robust Augmented Reality Registration”, In Proceedings of IEEE Virtual Reality, pp 71-78, March 2001.

[25] J. Corrales, F. Candelas, F. Torres, “Hybrid Tracking of Human Operators using IMU/UWB Data Fusion by a Kalman Filter”, In

Proceedings of IEEE Human-Robot Interaction International Conference, pp 193-200, March 2008.

[26] S. Sun, X. Meng, L. Ji, J. Wu, “Adaptive Sensor Data Fusion in Motion Capture”, In Proceedings of IEEE Information Fusion Conference, pp. 1-8, July 2010.

[27] D. Roetenberg, P. J. Slycke, P. H. Veltink, “Ambulatory Position and Orientation Tracking Fusing Magnetic and Inertial Sensing”, IEEE Transactions on Biomedical Engineering, Vol.54, Issue 5, pp. 883-890, May 2007.

[28] R. Mavani, P. Ponnammal, “MEMS Accelerometer Based 3D Mouse and Handwritten Recognition System”, International Journal of Innovative Research in Computer and Communication Engineering, Vol. 2, Issue 3, pp. 3333-3339, March 2014.

[29] Y. Xue, D. Yin, H. Sun, S. Zheng, “Design and Development of Portable Handheld Terminal for Monitor System Based on Embedded Technology”, Applied Mechanics and Materials, pp. 2796-2799, December 2012.

[30] Y. Lin, I. Jan, P. Ko, Y. Chen, “A Wireless PDA-Based Physiological Monitoring System for Patient Transport”, IEEE Transactions on Information Technology in Biomedicine, Vol. 8, Issue 4, pp. 439-447, December 2004.

[31] T. Kao, C. Lin, J. Wang, “Development of a Portable Activity Detector for Daily Activity Recognition”, In Proceedings of IEEE International Symposium on Industrial Electronics, pp. 115-120, July 2009.

[32] N. Pranathi, S. Ahmed, “Tri-Axis Motion Detection using MEMS for Unwired Mouse Navigation System in the Future Generation Machines”, International Journal of Advanced Research in Computer and Communication Engineering, Vol. 2, Issue 9, September 2013.

Appendix Ⅰ

Source code for portable monitoring system for home and office security (written in python)

Test.py

```
import smbus
import time
import smtplib

bus=smbus.SMBus(1)
addr=0x68
bus.write_byte_data(addr,0x6B,0x00)
import time
import smtplib, os
# Send an HTML email with an embedded image and a plain text message for
# email clients that don't want to display the HTML.

from email.MIMEMultipart import MIMEMultipart
from email.MIMEText import MIMEText
from email.MIMEImage import MIMEImage

#function to take a picture
def cam():
    import subprocess
    grab_cam = subprocess.Popen("sudo fswebcam -r 352x288 -d /dev/video0 -q
/home/pi/test.jpg", shell=True) #replace as necessary
    grab_cam.wait()

    #print "Acquiring image file...."

# Define these once; use them twice!
strFrom = 'duliqiang8@gmail.com'
strTo = 'liqiangd@mtu.edu'
username = 'duliqiang8@gmail.com'
password = '10woaiwoj07'
def send_mail():
    # Create the root message and fill in the from, to, and subject headers
    msgRoot = MIMEMultipart('related')
    msgRoot['Subject'] = 'Alert!!!'
    msgRoot['From'] = strFrom
```

```
        msgRoot['To'] = strTo
        msgRoot.preamble = 'This is a multi-part message in MIME format.'

        # Encapsulate the plain and HTML versions of the message body in an
        # 'alternative' part, so message agents can decide which they want to display.
        msgAlternative = MIMEMultipart('alternative')
        msgRoot.attach(msgAlternative)

        msgText = MIMEText('This is the alternative plain text message.')
        msgAlternative.attach(msgText)

        # We reference the image in the IMG SRC attribute by the ID we give it
below
        msgText = MIMEText('<b>Somebody might <i>break in</i> and here is</b>
an image of him.<br><img src="cid:image1"><br>', 'html')
        msgAlternative.attach(msgText)

        # This example assumes the image is in the current directory
        fp = open('test.jpg', 'rb')
        msgImage = MIMEImage(fp.read())
        fp.close()

        # Define the image's ID as referenced above
        msgImage.add_header('Content-ID', '<image1>')
        msgRoot.attach(msgImage)

        # Send the email (this example assumes SMTP authentication is required)
        smtp = smtplib.SMTP('smtp.gmail.com:587')
        smtp.starttls()
        smtp.login(username,password)
        smtp.sendmail(strFrom, strTo, msgRoot.as_string())
        smtp.quit()
        print "Email sent"

print "Starting monitoring"
while True:
        ax_h=bus.read_byte_data(addr,0x3B)
        ax_l=bus.read_byte_data(addr,0x3C)
        ay_h=bus.read_byte_data(addr,0x3D)
        ay_l=bus.read_byte_data(addr,0x3E)
        az_h=bus.read_byte_data(addr,0x3F)
        az_l=bus.read_byte_data(addr,0x40)
        gx_h=bus.read_byte_data(addr,0x43)
        gx_l=bus.read_byte_data(addr,0x44)
```

```
gy_h=bus.read_byte_data(addr,0x45)
gy_l=bus.read_byte_data(addr,0x46)
gz_h=bus.read_byte_data(addr,0x47)
gz_l=bus.read_byte_data(addr,0x48)
if ax_h&0x80 == 128:
        ax= -0x8000+(((ax_h&0x7F)<<8)|ax_l)
else:
        ax=(ax_h<<8)|ax_l
if ay_h&0x80 == 128:
        ay= -0x8000+(((ay_h&0x7F)<<8)|ay_l)
else:
        ay=(ay_h<<8)|ay_l
if az_h&0x80 == 128:
        az= -0x8000+(((az_h&0x7F)<<8)|az_l)
else:
        az=(az_h<<8)|az_l
if gx_h&0x80 == 128:
        gx= -0x8000+(((gx_h&0x7F)<<8)|gx_l)
else:
        gx=(gx_h<<8)|gx_l
if gy_h&0x80 == 128:
        gy= -0x8000+(((gy_h&0x7F)<<8)|gy_l)
else:
        gy=(gy_h<<8)|gy_l
if gz_h&0x80 == 128:
        gz= -0x8000+(((gz_h&0x7F)<<8)|gz_l)
else:
        gz=(gz_h<<8)|gz_l
if abs(gy)>3000:
        cam()
        send_mail()
#print ax,ay,az,gx,gy,gz
time.sleep(0.25)
```

Appendix Ⅱ

Source code for 3D mouse (written in C)

```
/*
 * main.c
 */
#include <msp430g2553.h>
#include <stdint.h>
#include "MPU6050.h"
#include "stdarg.h"

#define   MPU6050Addr 0x68

uint8_t *PRx,*PTx;
uint8_t RXByteCtr=0;
uint8_t TXByteCtr=0;
uint8_t RX=0;
uint8_t buffer[14];
uint8_t a=0x01;
uint8_t b=0x03;
uint8_t int_0 = 0x30;
uint8_t int_1 = 0x31;
uint8_t int_2 = 0x32;
uint8_t int_3 = 0x33;
uint8_t int_4 = 0x34;
uint8_t int_5 = 0x35;
uint8_t int_6 = 0x36;
uint8_t int_7 = 0x37;
uint8_t int_8 = 0x38;
uint8_t int_9 = 0x39;


void initUart(void)
{
        UCA0CTL1 |= UCSSEL_2;                     // Use SMCLK
        UCA0BR0 = 104;                            // 1MHz 9600
        UCA0BR1 = 0;                              // 1MHz 9600
        UCA0MCTL = UCBRS0;                  // Modulation UCBRSx = 1
        P1SEL = BIT1 + BIT2 ;                 // P1.1 = RXD, P1.2=TXD
        P1SEL2 = BIT1 + BIT2 ;              // P1.1 = RXD, P1.2=TXD
        UCA0CTL1 &= ~UCSWRST;     // **Initialize USCI state machine**
```

```
        IE2 |= UCA0TXIE;
}

void Init_i2c(uint8_t devAddr)
{
 UCB0CTL1 |= UCSWRST; // Enable SW reset
 UCB0CTL0 = UCMST + UCMODE_3 + UCSYNC; // I2C Master, synchronous mode
 UCB0CTL1 = UCSSEL_2 + UCSWRST; // Use SMCLK, keep SW reset
 UCB0BR0 = 10; // fSCL = 1Mhz/10 = ~100kHz
 UCB0BR1 = 0;
 P1SEL =  BIT6 + BIT7; // Assign I2C pins to USCI_B0 // Assign Uart pins to USCI_A0
 P1SEL2 = BIT6 + BIT7; // Assign I2C pins to USCI_B0 // Assign Uart pins to USCI_A0
 UCB0I2CSA = devAddr; // Slave Address is 069h
 UCB0CTL1 &= ~UCSWRST; // **Initialize USCI state machine**
 IE2 |= UCB0RXIE + UCB0TXIE; // Enable RX and TX interrupt
}
uint8_t intToAscii(uint8_t value) {
        if(value == 0) {
                return int_0;
        }
        else if(value == 1) {
                return int_1;
        }
        else if(value == 2) {
                return int_2;
        }
        else if(value == 3) {
                return int_3;
        }
        else if(value == 4) {
                return int_4;
        }
        else if(value == 5) {
                return int_5;
        }
        else if(value == 6) {
                return int_6;
        }
        else if(value == 7) {
                return int_7;
        }
```

```
        else if(value == 8) {
                return int_8;
        }
        else if(value == 9) {
                return int_9;
        }
        else if(value == '-') {
                        return 0x2D;
                }
        else if(value == '+') {
                                return 0x2B;
                        }
        else return 0x00;
}
void serialWrite(uint8_t num) {
 UCA0TXBUF = intToAscii(num);
 while (!(IFG2&UCA0TXIFG)); // USCI_A0 TX buffer ready?

}

void serialWriting(int16_t num)
{
 int i;
 int16_t num_send[6];
 int16_t numTemp;
 IE2 &= ~(UCA0TXIE);
 IE2 &= ~(UCA0RXIE);
 if (num < 0)
 {
      num=0-num;
      num_send[0]='-';
 }
 else
      num_send[0]='+';
      num_send[1] = num/10000;    // extract 5th digit
      numTemp = num % 10000;          // get remaining 4
      num_send[2] = numTemp/1000; // extract 4th digit
      numTemp = numTemp % 1000;    // get remamining 3
      num_send[3] = numTemp/100;      // extract 3th digit
      numTemp = numTemp % 100;       // get remaining 2
      num_send[4] = numTemp/10; // extract 2th digit
      num_send[5] = numTemp % 10; // extract 1th digit
      TXByteCtr=5;
      PTx=(uint8_t *)num_send;
```

```
    IE2 |= UCA0TXIE;
    __bis_SR_register(CPUOFF + GIE);            // Enter LPM0 w/ interrupts

        for(i = 0 ; i <= 5 ; i++)
        serialWrite(num_send[i]);// send each digit as one byte
}



void transmitIni(uint8_t regAddr)
{
 while (UCB0CTL1 & UCTXSTP); // Ensure stop condition got sent
 UCB0CTL1 |= UCTR + UCTXSTT; // I2C start condition with UCTR flag for transmit
 while((IFG2 & UCB0TXIFG) == 0); //UCB0TXIFG is set immidiately
 UCB0TXBUF = regAddr; //write registerAddr in TX buffer
 while((IFG2 & UCB0TXIFG) == 0); // wait until TX buffer is empty and transmitted
}

void ReadBytes(uint8_t devAddr, uint8_t regAddr, uint8_t length, uint8_t *data)
{
 _DINT();
 RX=1;
 //Init_i2c(devAddr);
 IE2 &= ~(UCB0TXIE);
 IE2 &= ~(UCB0RXIE);
 transmitIni(regAddr);
 IE2 |= UCB0RXIE;//+ UCB0TXIE; // Enable RX and TX interrupt
 PRx = (uint8_t *)data;
 RXByteCtr = length;
 while (UCB0CTL1 & UCTXSTP);                 // Ensure stop condition got sent
 UCB0CTL1 &= ~UCTR ; // Clear I2C TX flag for receive
 UCB0CTL1 |= UCTXSTT;                        // I2C TX, start condition
 //while (UCB0CTL1 & UCTXSTT); // Start condition sent?
 __bis_SR_register(CPUOFF + GIE);            // Enter LPM0 w/ interrupts
}

void WriteBytes(uint8_t devAddr, uint8_t regAddr, uint8_t length, uint8_t *data)
{
 _DINT();
 RX=0;
 //Init_i2c(devAddr);
```

```
 IE2 &= ~(UCB0TXIE);
 IE2 &= ~(UCB0RXIE);
 transmitIni(regAddr);
 IE2 |= UCB0TXIE; // Enable RX and TX interrupt
 TXByteCtr = length;
 PTx = (uint8_t *)data;
 while (UCB0CTL1 & UCTXSTP);                    // Ensure stop condition got
sent
//     UCB0CTL1 |= UCTR + UCTXSTT;                    // I2C TX, start
condition
//     while (UCB0CTL1 & UCTXSTT); // Start condition sent?
 __bis_SR_register(CPUOFF + GIE);              // Enter LPM0 w/ interrupts
}

#pragma vector = USCIAB0TX_VECTOR
__interrupt void USCIAB0TX_ISR(void)
{
 if((IFG2&UCA0TXIFG)&&(IE2&UCA0TXIE))                    //UART
 {
      if (TXByteCtr)
           {
                UCA0TXBUF = intToAscii(*PTx++);
                TXByteCtr--;
           }
      else
           {
                IE2 &= ~UCA0TXIE;             // Disable USCI_A0 TX interrupt
                IFG2 &= ~UCB0TXIFG;           // Clear USCI_B0 TX int flag
                __bic_SR_register_on_exit(CPUOFF);          // Exit LPM0
           }
 }
 if((IE2&UCB0TXIE)||(IE2&UCB0RXIE))
 {
   if(RX == 1)
   {                                          // Master Recieve?
    RXByteCtr--;                             // Decrement RX byte counter
    if (RXByteCtr)
       {
            *PRx++ = UCB0RXBUF;       // Move RX data to address PRxData
       }
    else
       {
            UCB0CTL1 |= UCTXSTP; // No Repeated Start: stop condition
            *PRx = UCB0RXBUF;       // Move final RX data to PRxData
```

```
            __bic_SR_register_on_exit(CPUOFF);          // Exit LPM0
        }
    }

    else
    {                                                   // Master Transmit
        if (TXByteCtr)                                  // Check TX byte counter
        {
            UCB0TXBUF = *PTx++;                         // Load TX buffer
            while((IFG2 & UCB0TXIFG) == 0); // wait until TX buffer is empty
and transmitted
            TXByteCtr--;                            // Decrement TX byte counter
        }
        else
        {
            UCB0CTL1 |= UCTXSTP;                    // I2C stop condition
            IFG2 &= ~UCB0TXIFG;                 // Clear USCI_B0 TX int flag
            __bic_SR_register_on_exit(CPUOFF);          // Exit LPM0
        }
    }
 }
}

void ReadByte(uint8_t devAddr, uint8_t regAddr,uint8_t* data)
{
 //Init_i2c(devAddr);
 //RX=1;
 IE2 &= ~(UCB0TXIE);
 IE2 &= ~(UCB0RXIE);
 transmitIni(regAddr);

 while (UCB0CTL1 & UCTXSTP);            // Ensure stop condition got sent
 UCB0CTL1 &= ~UCTR ; // Clear I2C TX flag for receive
 UCB0CTL1 |= UCTXSTT;                   // I2C TX, start condition
 //while (UCB0CTL1 & UCTXSTT); // Start condition sent?

 UCB0CTL1 |= UCTXSTP; // I2C stop condition
 *data = UCB0RXBUF;

 //while((IFG2 & UCB0RXIFG) == 0); // Wait until data read
 IE2 |= UCB0RXIE + UCB0TXIE; // Enable RX and TX interrupt
}

void WriteByte(uint8_t devAddr, uint8_t regAddr, uint8_t data)
```

```
{
 //Init_i2c(devAddr);
 IE2 &= ~(UCB0TXIE);
 IE2 &= ~(UCB0RXIE);
 transmitIni(regAddr);

 UCB0TXBUF = data; //write registerAddr in TX buffer
 while((IFG2 & UCB0TXIFG) == 0); // wait until TX buffer is empty and transmitted
 UCB0CTL1 |= UCTXSTP; // I2C stop condition
 IFG2 &= ~UCB0TXIFG;
 IE2 |= UCB0RXIE + UCB0TXIE; // Enable RX and TX interrupt
}


void getMotion6(int16_t* ax, int16_t* ay, int16_t* az, int16_t* gx, int16_t* gy, int16_t* gz)          //<! Function that retrieves the 6-axis motion from MPU6050/6000
{

 ReadBytes(MPU6050Addr, MPU6050_RA_ACCEL_XOUT_H, 14, buffer);
 *ax = (((int16_t)buffer[0]) << 8) | buffer[1];
 *ay = (((int16_t)buffer[2]) << 8) | buffer[3];
 *az = (((int16_t)buffer[4]) << 8) | buffer[5];
 *gx = (((int16_t)buffer[8]) << 8) | buffer[9];
 *gy = (((int16_t)buffer[10]) << 8) | buffer[11];
 *gz = (((int16_t)buffer[12]) << 8) | buffer[13];
}
void initializeIMU()
                                    //<! Function that setups IMU. This function can be altered by user.
{

 WriteBytes(MPU6050Addr, MPU6050_RA_PWR_MGMT_1,1, &a);

 WriteBytes(MPU6050Addr, MPU6050_RA_GYRO_CONFIG,1, &b);

 WriteBytes(MPU6050Addr, MPU6050_RA_ACCEL_CONFIG, 1,&a);

}


void main(void) {
 int16_t ax,ay,az,gx,gy,gz;
```

```
 int16_t
gxFiltered=0,gyFiltered=0,gzFiltered=0,gxPrevious=0,gyPrevious=0,gzPrevious=
0;
 int16_t
axFiltered=0,ayFiltered=0,azFiltered=0,axPrevious=0,ayPrevious=0,azPrevious=0
;
 WDTCTL = WDTPW + WDTHOLD; // Stop WDT
 BCSCTL1 = CALBC1_1MHZ; // Set DCO to 1Mhz
 DCOCTL = CALDCO_1MHZ;
 P1DIR=0x01;
 P1OUT=0;
 Init_i2c(MPU6050Addr);
 initializeIMU();
 getMotion6( &ax, &ay, &az, &gx, &gy, &gz);
 axFiltered=ax;
 ayFiltered=ay;
 azFiltered=az;
 gxFiltered=gx;
 gyFiltered=gy;
 gzFiltered=gz;
 while(1)
 {

      P1OUT=1;
      _delay_cycles(10);
      Init_i2c(MPU6050Addr);
      initializeIMU();

      getMotion6( &ax, &ay, &az, &gx, &gy, &gz);
      _delay_cycles(10);
      axPrevious = axFiltered;
      axFiltered = axPrevious + (0.3 * (ax - axPrevious));
      ayPrevious = ayFiltered;
      ayFiltered = ayPrevious + (0.3 * (ay - ayPrevious));
      azPrevious = azFiltered;
      azFiltered = azPrevious + (0.3 * (az - azPrevious));
      gxPrevious = gxFiltered;
      gxFiltered = gxPrevious + (0.6 * (gx - gxPrevious));
      gyPrevious = gyFiltered;
      gyFiltered = gyPrevious + (0.6 * (gy - gyPrevious));
      gzPrevious = gzFiltered;
      gzFiltered = gzPrevious + (0.6 * (gz - gzPrevious));
      UCB0CTL1 = 0x01; // Enable SW reset
      UCB0CTL0 = 0x01;
```

```
        initUart();
        serialWriting(axFiltered);

        serialWriting(ayFiltered);

        serialWriting(azFiltered);

        serialWriting(gxFiltered);

        serialWriting(gyFiltered);

        serialWriting(gzFiltered);

        //UCA0TXBUF = 0x0A;
        //while (!(IFG2&UCA0TXIFG)); // USCI_A0 TX buffer ready?
        _delay_cycles(10);

        UCA0CTL1 =0x01;
        //P1OUT=0;
        _delay_cycles(10);
 }
}
```

PC source code for 3D mouse

```
// ComTestDlg.cpp : implementation file
//

#include "stdafx.h"
#include "ComTest.h"
#include "ComTestDlg.h"
#include "afxdialogex.h"
#include <cmath>

#ifdef _DEBUG
#define new DEBUG_NEW
#endif
static int gllen;
unsigned char count;
static int number=36;
float RADIANS_TO_DEGREES = 180/3.14159;
float DEGREES_TO_RADIANS = 3.14159/180;
```

```
static int tran;
static char ax[6],ay[6],az[6];
static char gx[6],gy[6],gz[6];
static float accel_x=0,accel_y=0,accel_z=0,gyro_x=0,gyro_y=0,gyro_z=0;
static unsigned long last_read_time=0;
static float		last_x_angle=0;	// These are the filtered angles
static float		last_y_angle=0;
static float		last_z_angle=0;
static float		last_gyro_x_angle=0;	// Store the gyro angles to compare drift
static float		last_gyro_y_angle=0;
static float		last_gyro_z_angle=0;
static float x=0,z=0;
static float vx=0,vz=0;
static int sx=0,sz=0;
float a_x=0,a_z=0;
static float angle_x=0;
static float angle_y=0;
static float angle_z=0;

void firsttimedata();
//void micemove(int a,int b);
void finaldata();
// CAboutDlg
// CAboutDlg dialog used for App About

class CAboutDlg : public CDialogEx
{
public:
 CAboutDlg();

// Dialog Data
 enum { IDD = IDD_ABOUTBOX };

 protected:
 virtual void DoDataExchange(CDataExchange* pDX);	// DDX/DDV support

// Implementation
protected:
 DECLARE_MESSAGE_MAP()
};

CAboutDlg::CAboutDlg() : CDialogEx(CAboutDlg::IDD)
{
```

```
}

void CAboutDlg::DoDataExchange(CDataExchange* pDX)
{
 CDialogEx::DoDataExchange(pDX);
}

BEGIN_MESSAGE_MAP(CAboutDlg, CDialogEx)
END_MESSAGE_MAP()


// CComTestDlg dialog



CComTestDlg::CComTestDlg(CWnd* pParent /*=NULL*/)
 : CDialogEx(CComTestDlg::IDD, pParent)
 , m_ReceiveData(_T(""))
{
 m_hIcon = AfxGetApp()->LoadIcon(IDR_MAINFRAME);
}

void CComTestDlg::DoDataExchange(CDataExchange* pDX)
{
 CDialogEx::DoDataExchange(pDX);
 DDX_Control(pDX, IDC_MSCOMM1, m_Com);
 DDX_Control(pDX, IDC_BUTTON1, m_OpenSerial);
 DDX_Control(pDX, IDC_BUTTON2, m_ReadData);
 DDX_Text(pDX, IDC_EDIT1, m_ReceiveData);
}

BEGIN_MESSAGE_MAP(CComTestDlg, CDialogEx)
 ON_WM_SYSCOMMAND()
 ON_WM_PAINT()
 ON_WM_QUERYDRAGICON()
END_MESSAGE_MAP()


// CComTestDlg message handlers

BOOL CComTestDlg::OnInitDialog()
{
 CDialogEx::OnInitDialog();
```

```
// IDM_ABOUTBOX
ASSERT((IDM_ABOUTBOX & 0xFFF0) == IDM_ABOUTBOX);
ASSERT(IDM_ABOUTBOX < 0xF000);

CMenu* pSysMenu = GetSystemMenu(FALSE);
if (pSysMenu != NULL)
{
	BOOL bNameValid;
	CString strAboutMenu;
	bNameValid = strAboutMenu.LoadString(IDS_ABOUTBOX);
	ASSERT(bNameValid);
	if (!strAboutMenu.IsEmpty())
	{
		pSysMenu->AppendMenu(MF_SEPARATOR);
		pSysMenu->AppendMenu(MF_STRING, IDM_ABOUTBOX,
strAboutMenu);
	}
}

//
SetIcon(m_hIcon, TRUE);			// Set big icon
SetIcon(m_hIcon, FALSE);		// set small icon
gllen=0;
m_Com.put_CommPort(5);//Com port 5
m_Com.put_PortOpen(TRUE);//open COM port
m_Com.put_RThreshold(2);
m_Com.put_InputMode(1);
m_Com.put_Settings(_T("9600,n,8,1"));//Baud rate 9600

return TRUE;
}

void CComTestDlg::OnSysCommand(UINT nID, LPARAM lParam)
{
if ((nID & 0xFFF0) == IDM_ABOUTBOX)
{
	CAboutDlg dlgAbout;
	dlgAbout.DoModal();
}
else
{
	CDialogEx::OnSysCommand(nID, lParam);
}
```

```
}

// If you add a minimize button to your dialog, you will need the code below
//    to draw the icon.    For MFC applications using the document/view model,
//    this is automatically done for you by the framework.

void CComTestDlg::OnPaint()
{
 if (IsIconic())
 {
      CPaintDC dc(this); // device context for painting

      SendMessage(WM_ICONERASEBKGND,
reinterpret_cast<WPARAM>(dc.GetSafeHdc()), 0);

      // Center icon in client rectangle
      int cxIcon = GetSystemMetrics(SM_CXICON);
      int cyIcon = GetSystemMetrics(SM_CYICON);
      CRect rect;
      GetClientRect(&rect);
      int x = (rect.Width() - cxIcon + 1) / 2;
      int y = (rect.Height() - cyIcon + 1) / 2;

      // Draw the icon
      dc.DrawIcon(x, y, m_hIcon);
 }
 else
 {
      CDialogEx::OnPaint();
 }
}

// The system calls this function to obtain the cursor to display while the user
drags
//    the minimized window.
HCURSOR CComTestDlg::OnQueryDragIcon()
{
 return static_cast<HCURSOR>(m_hIcon);
}

BEGIN_EVENTSINK_MAP(CComTestDlg, CDialogEx)
 ON_EVENT(CComTestDlg,                IDC_MSCOMM1,                1,
CComTestDlg::OnCommMscomm1, VTS_NONE)
END_EVENTSINK_MAP()
```

```
void CComTestDlg::OnCommMscomm1()
{
 VARIANT variant_inp;
 int i;
 static float h=1280/2,v=728/2;
       COleSafeArray safearray_inp;
       LONG len,k;
       BYTE rxdata[2048];
       CString strtemp;
       if(m_Com.get_CommEvent() == 2)         {
       variant_inp= m_Com.get_Input();
safearray_inp= variant_inp;//
len= safearray_inp.GetOneDimSize(); //
for(k = 0; k < len; k++)
         safearray_inp.GetElement(&k, rxdata+k);//
//             UpdateData(true);
              for(k = 0;k<len;k++) //
              {
                    tran=*(int*)(rxdata+k);
             BYTE bt = *(char*)(rxdata+k); //
             //strtemp.Format(_T("%d"), bt-48);      //
             //m_ReceiveData+=strtemp;

             if (-1<number%36 && number%36<6)//
                   ax[number%36]=bt;//
             if (5<number%36 && number%36<12)
                   ay[(number%36)-6]=bt;//
             if (11<number%36 && number%36<18)
                   az[(number%36)-12]=bt;//
if (17<number%36 && number%36<24)
                   gx[(number%36)-18]=bt;//
             if (23<number%36 && number%36<30)
                   gy[(number%36)-24]=bt;//
             if (29<number%36 && number%36<36)
                   gz[(number%36)-30]=bt;//
             number++;
             if (number%36==0)//
             {
                   finaldata();
                   strtemp.Format(_T("%f,%f"), gyro_z/134,gyro_y/134);
                   m_ReceiveData+=strtemp;
                   m_ReceiveData+="\r\n";
```

```
                for(i=0;i<2000;i++)//mouse curser move function
                    {

                        h=h+(float)z/2000;//coordinate
                        v=v+(float)x/2000;
                        if(h>1280)
                            h=1280;
                        if(v>1280)
                            v=1280;
                        if(h<0)
                            h=0;
                        if(v<0)
                            v=0;
                        SetCursorPos(h,v);//move mouse curser
                    }

            }
        }
SetDlgItemText(IDC_EDIT1,m_ReceiveData);
 }
}

void set_last_read_angle_data(unsigned long time, float x, float y, float z) {
   last_read_time = time;
   last_x_angle = x;
   last_y_angle = y;
   last_z_angle = z;

}


inline unsigned long get_last_time() {return last_read_time;}
inline float get_last_x_angle() {return last_x_angle;}
inline float get_last_y_angle() {return last_y_angle;}
inline float get_last_z_angle() {return last_z_angle;}
void firsttimedata()//MCU {
 int al,am,an,gl,gm,gn;
 unsigned static long t_now = 0;
 t_now += 0.04;
 if((ax[0]-48)!=-3)al=1;
 else al=-1;
 if((ay[0]-48)!=-3)am=1;
 else am=-1;
 if((az[0]-48)!=-3)an=1;
```

```
 else an=-1;
 if((gx[0]-48)!=-3)gl=1;
 else gl=-1;
 if((gy[0]-48)!=-3)gm=1;
 else gm=-1;
 if((gz[0]-48)!=-3)gn=1;
 else gn=-1;
 accel_x=al*((ax[1]-48)*10000+(ax[2]-48)*1000+(ax[3]-48)*100+(ax[4]-48)*10
+ax[5]-48);
 accel_y=am*((ay[1]-48)*10000+(ay[2]-48)*1000+(ay[3]-48)*100+(ay[4]-48)*10
+ay[5]-48);
 accel_z=an*((az[1]-48)*10000+(az[2]-48)*1000+(az[3]-48)*100+(az[4]-48)*10+
az[5]-48);
 gyro_x=gl*((gx[1]-48)*10000+(gx[2]-48)*1000+(gx[3]-48)*100+(gx[4]-48)*10
+gx[5]-48)/131;
 gyro_y=gm*((gy[1]-48)*10000+(gy[2]-48)*1000+(gy[3]-48)*100+(gy[4]-48)*1
0+gy[5]-48)/131;
 gyro_z=gn*((gz[1]-48)*10000+(gz[2]-48)*1000+(gz[3]-48)*100+(gz[4]-48)*10
+gz[5]-48)/131;
 float accel_angle_x = atan(-1*accel_x/sqrt(pow(accel_y,2) +
pow(accel_z,2)))*RADIANS_TO_DEGREES;
      float accel_angle_y = atan(accel_y/sqrt(pow(accel_x,2) +
pow(accel_z,2)))*RADIANS_TO_DEGREES;
 float accel_angle_z = 0;

 // Compute the (filtered) gyro angles
 float dt =0.04;//(t_now - get_last_time());
      float gyro_angle_x = gyro_x*dt + get_last_x_angle();
      float gyro_angle_y = gyro_y*dt + get_last_y_angle();
      float gyro_angle_z = gyro_z*dt + get_last_z_angle();
 /*
 // Compute the drifting gyro angles
 float unfiltered_gyro_angle_x = gyro_x*dt + get_last_gyro_x_angle();
 float unfiltered_gyro_angle_y = gyro_y*dt + get_last_gyro_y_angle();
 float unfiltered_gyro_angle_z = gyro_z*dt + get_last_gyro_z_angle();
 */
 // Apply the complementary filter to figure out the change in angle - choice of
alpha is
   // estimated now.    Alpha depends on the sampling rate...
 float alpha = 0.96;
 angle_x = alpha*gyro_angle_x + (1.0 - alpha)*accel_angle_x;
 angle_y = alpha*gyro_angle_y + (1.0 - alpha)*accel_angle_y;
 angle_z = gyro_angle_z;    //Accelerometer doesn't give z-angle
```

```
 set_last_read_angle_data(t_now, angle_x, angle_y, angle_z);
}
void finaldata(){
 firsttimedata();    //
//ax=angle_x;
 //az=angle_y;
 //accel and gyro fusion
 /*
 a_x=((accel_y-16384*sin(angle_y*DEGREES_TO_RADIANS))/cos(angle_y*D
EGREES_TO_RADIANS))-accel_z*sin(angle_z*DEGREES_TO_RADIANS);
 a_z=((accel_z-16384*cos(angle_z*DEGREES_TO_RADIANS))/cos(angle_z*D
EGREES_TO_RADIANS))-accel_y*sin(angle_y*DEGREES_TO_RADIANS);
 a_x/=16384;
 a_z/=16384;
 if(a_x>0.1||a_x<-0.1)
       a_x=a_x;
 else
       a_x=0;
 if(a_z>0.1||a_z<-0.1)
       a_z=a_z;
 else
       a_z=0;
 if(a_x==0)
       vx=0;
 if(a_z==0)
       vz=0;
 vx=(vx+a_z*15);
 vz=(vz+a_x*15);
 x=(vx+0.5*a_z*15);
 z=(vz+0.5*a_x*15);
 */
 /*
 //only gyro method

 float ax=0,az=0;
 ax=gyro_y;
 az=gyro_z;

 if(ax>5||ax<-5)
       ax=ax;
 else
       ax=0;
 if(az>5||az<-5)
       az=az;
```

```
else
      az=0;
if(ax==0)
      vx=0;
if(az==0)
      vz=0;
vx=(vx+ax/110);
vz=(vz-az/110);
x=(vx+0.5*ax/110)*30;
z=(vz-0.5*az/110)*30;

*/

float gx=0,gz=0,ay,az,gxold=0,gzold=0;
gx=gyro_y;
gz=gyro_z;
ay=accel_y/16384;
az=accel_z/16384-1;
if(gx>5||gx<-5)
      gx=gx;
else
      gx=0;
if(gz>5||gz<-5)
      gz=gz;
else
      gz=0;
if(gx==0)
      vx=0;
if(gz==0)
      vz=0;
if((ay>0.1||ay<-0.1)&&(ay<0.3||ay>-0.3))
      ay=ay-0.1;
else
      ay=0;
if((az>0.1||az<-0.1)&&(az<0.3||az>-0.3))
      az=az-0.1;
else
      az=0;
if(gx==0&&gz==0)
{ay=0;az=0;}
//vx=(vx+ax/110);
//vz=(vz-az/110);
x=(gx/60+0.5*az/50)*70;
z=-(gz/60+0.5*ay/50)*70;
```

```
//x=(gx/60+0.5*(gx-gxold)/50)*70;
//z=-(gz/60+0.5*(gz-gzold)/50)*70;
gxold=gx;
gzold=gz;
//only accelerate method
/*
float ax=0,ay=0;
ay=accel_y/16384;
ax=accel_x/16384;
if((ay>0.1||ay<-0.1)&&(ay<0.3||ay>-0.3))
      ay=ay-0.1;
else
      ay=0;
if((ax>0.1||ax<-0.1)&&(ax<0.3||ax>-0.3))
      ax=ax-0.1;
else
      ax=0;
//    if(ax>ay)
//          ay=0;
//    else
//          ax=0;
 if(ax==0)
      vx=0;
 if(ay==0)
      vz=0;

 //ax=0;
 vx=(vx+ax*25);
 vz=(vz+ay*25);
 x=(vx+0.5*ax*25);
 z=(vz+0.5*ay*25);
 */
}
```