ABSTRACT

With the concept of wearable device become more popular, both industries and individual developers are trying to seek a new type of operation to take the place of traditional computer mice. Air mouse is one of the most popular solution among all the existing ideas. However, some completed products which are being sold cost too much comparing traditional optical mice. This project is trying to find a possibility to build an air mouse prototype based on Arduino, accelerometer/gyroscope module and Bluetooth module. Alternative solutions were tested and compared to find the most suitable one to achieve this project. As the result, main function of air mouse can be achieved and the cost of the prototype is much lower than the existing products. However, some advanced function may be difficult to be achieved due to the limitation of breakout boards that cannot meet the requirement of size. And Algorithms are still needed to be improved.

ACKNOWLEDGMENTS

First I would like to express my respect and gratitude to my supervisor Dr. Christian Urban. He consistently gave me helpful suggestions not only to keep the project going at the right direction, also encouraged me to seek more ideas when problems occurred. His countless help was the mostly appreciated and was the foundation of this project.

I would also like to express my profound gratitude to my parents and my friends. Without their invaluable support and encouragement, I would never have been able to overcome difficulties confidently.

Table of Content

List of Figure

I. Introduction

1.1 Overview

A computer mouse is an input device (hand control) used for controlling the cursor whose position is determined by detecting two-dimensional motion on a surface. The motion of the mouse can be translated to PC and let the cursor have corresponding reaction which allows users to have graphical interface instead of purely keyboard control.

Invented by Douglas Engelbart in 1964, the first mouse consists of a wooden shell, circuit board and two metal wheels contacting the surface where it was being used on. Although it is different from those mice people normally use nowadays, both in appearance and technologies, the way of controlling is almost the same, even optical mice which became the mainstream choice instead of mechanical mice (ball mice) on desktop PC about 20 years ago.

In 2007, Logitech introduces the Logitech® MX Air™ Mouse which can work both on desk and in the air. It can be said as the first commercial mouse utilizing gyroscope to detect rotary motion therefore to control the cursor. [1] Considering that, in some critical situation, "traditional" wired mouse and touch pad cannot meet the needs, e.g. giving presentations, and some users may not be able to use these kinds of methods. Since then, industries began to develop new type of interfacing controllers. One of the most popular solution is called "air mouse", which uses accelerometer or gyroscope to detect rotation purely without any contact with any surfaces.

With the development of microcontrollers and integrated circuits, both economic and technical barrier of developing a project or prototype by individuals became much lower for individual developers or hobbyists. Therefore both individual and industries are able to seek new ways for computer mouse with similar or even higher compatibility. The increasing number of communities where developers can share their ideas and have discussions, also provides a positive and encouraging atmosphere. Effected by these factors above, many projects relating to wearable air mouse can be found on Kickstarter. [2]

1.2 Objectives

Although most of the relative products are still at the stage of seeking financial supports, some completed products can already be found being sold on Amazon, such as *Mycestro* and *Myo Armband*. However, the price of these devices exceed over ￡100 which is much higher than the price of a normal laser mouse, which is the main problem that makes them less competitive and cannot be accepted by customers who are interested in or even really need it, especially comparing to the traditional optical mouse that normally costs ￡10, some are even less. The objective of this project is trying to give a low-cost solution of an air mouse based on Arduino using gyroscope and Bluetooth module.

1.3 Aims

The aims of this project are to find the possibility of building a low-cost an air mouse based on Arduino, and then to develop a prototype. These aims can be separated more specifically as following:

1). Read the datasheet of the accelerometer-gyroscope module and test it based on the datasheet and corresponding open-source test code.

2). Read the datasheet of the Bluetooth module and test it based on the datasheet and corresponding open-source test code.

3). Combine two components above together and write code to let the Arduino transmit the data from accelerometer-gyroscope module to PC through Bluetooth.

4). Read relative literatures and tutorials about translating the data collected to the parameters that controls the motion of the cursor.

5). Improve the hardware and connections to build an air mouse prototype.

1.4 **Organization**

Chapter 2 (Background) describes an overview of each component of this project as well as the construction, followed by some essential concepts It also provides some alternative options in both devices and coding, then the advantages and disadvantages will be discussed.

Chapter 3 (Technical Specification) provides the devices chosen for this project and some techniques (e.g. coding language, the platform chosen in this project etc.).

Chapter 4 (Methodologies) presents the details of the implementation of this project. This chapter will be divided into several parts according to the structure of whole project. Some test examples will be explained in order to make the description clearer and more detailed.

Chapter 5 (Conclusions and Improvements) evaluates the project objectives, and some problems occurred during the development which are not completely fixed and may improve the performance of the device if solved. Further tasks will be described at the end of this chapter.

II. Background

2.1 Overview

The previous chapter gives a brief history of the development of pointer controller (mice). It also gives a basic introduction of the principles that how does a traditional mouse work. In this chapter, the working theory will be described in a more detailed way, which makes the structure of this project clearer. And it also gives a background introduction of techniques and equipment that used in this project. To explain matters clearly, some pictures from other sources are used.

2.2 Working Principles of Pointer Controllers (Computer Mouse)

Regardless of the old types of mouse, there are mainly two types of mouse that are the most successful and widely used products during the history of mouse, which are mechanical mouse and optical mouse respectively. And it is obvious that optical mice have taken the place of mechanical mouse and become the most popular and common device in the market. By researching the working principles of these two types of mouse, especially the relationship between human motion and the cursor motion, it will be easier to figure out how does a wearable air mouse work and the essential structure of the device.

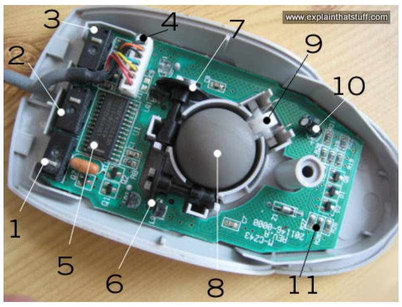


Figure 1 structure of a "ball mouse", with top cover removed [3]

As shown in Figure 1 above, modern mechanical mouse, also called "ball mouse", usually consists several main parts: clicking buttons (1 and 3), analog-to-digital converter (5), wheels on x-axis (6), wheels on y-axis (7), heavy rubber ball (8). When moving the mouse on a surface, the rubber ball will roll due to the force of friction (that is the reason why the balls are made of rubber and are heavy). This rotation can be detected by the two wheels in two axes. And some springs (9) are used for pushing the rubber ball which guarantee the ball is against the wheels firmly so that rotations can be detected precisely.

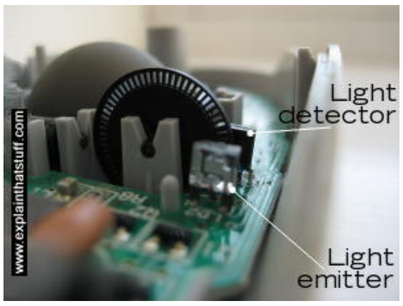


Figure 2 the structure of the wheel [4]

Figure 2 shows the detailed structure of one wheel. It can be seen that there are many spokes distributed evenly on the wheel. There also a light emitter and detector located on the different side of the wheel. As for converting the rotation to a measurable analog value, it is a simple but clever method. The light detector receives continuous infrared light beam when the rubber ball is motionless in its direction. When the mouse is moved, infrared light beam will be divided into several segments. And the number of light segments is relating to the velocity in this direction, the velocity then can be measured by counting the number of light segments received by the detector. These velocity data in two axes will be converted to digital signals then sent to PC which will be used to control the cursor.

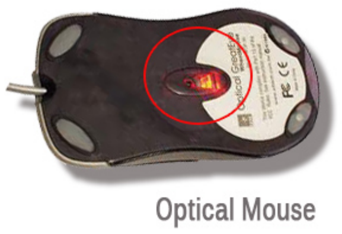


Figure 3 optical mouse pictured from bottom [5]

Unlike mechanical mice, optical mice work in a different way. As Figure 3 shows, instead of using rubber ball, the function of capturing motions is achieved by using a red light-emitting-diode (LED) and a light detecting sensor. When the mouse starts working, LED emits light to the surface and the photocell, which usually is a complementary metal-oxide semiconductor, detects the light which is reflected to it. A lens is also applied to the photocell to magnify the light which can improve the reaction and precision. After receiving the reflected light, the corresponding signal will be sent the to a DSP (digital signal processor) to analyze the change of the pattern of the reflected light. Then the motion including directions and velocities on each axis will be determined. Since the operation of DSP approximates million times per second, optical mouse can provide more precise and higher sensitivity performance than mechanical mouse. [6]

According to the research of the working principle of two mainstream mice, the structure of a computer mouse can be clarified. It generally contains three main parts: 1). detecting and measuring the human physical motions. 2). Converting the data of motions to corresponding parameters usually in two dimensions e.g. velocities on each axis or coordinates. 3). Using these parameters to determine the motion of the cursor. The procedures of the project are based on this structure.

2.3 Arduino

Arduino is an open-source electronics platform based on easy-to-use hardware and software. [7] The first prototype of Arduino was designed and developed in 2003 which is for those students without background in electronics and programming. Due to its simple, friendly and accessible user experience, Arduino is popular with the public from beginners to experienced engineers. After more than ten years' development, Arduino has become a well-developed product with a variety of types to meet different needs and challenges, and its own worldwide open-source community where thousands of ideas were shared.

The most attractive properties of Arduino are shown below: [8]

• Low Cost - Arduino boards are relatively inexpensive comparing to other mainstream microcontroller platforms. Even pre-assembled Arduino modules cost less than $50.

• Cross-platform - The Arduino Software (IDE) can run on Windows, Mac OS, and Linux operating systems.

• Simple for coding - The Arduino Software (IDE) is easy to understand for beginners, both in structure and syntax, and flexible enough for advanced users.

- Open source and extensible - The Arduino software is published as open source tools, available for extension through libraries in other language.
- High compatibility for hardware – With easy-to-sue GPIO and IDE, Arduino supports different kinds of circuits which gives the opportunities for industries or personal users to develop different kinds of modules to extend the functions of Arduino.

2.4 **Wearable Devices**

Based on the highly-integrated devices fabrication and more efficient wireless communication technologies, wearable devices have become a common part in daily life which is also a new popular market with high speed growing for device manufacturers. Relying on short-range wireless systems such as Bluetooth or local Wi-Fi setups, wearable devices are often used for tracking a user's vital signs or pieces of data related to health, fitness, and location etc. [9]

As the main characteristic, wearable devices reduce most of hands-on operations that alleviates users from the need to constantly draw their smartphone and check. It also provides opportunities for users to receive visual assistance in real-time while preforming a certain action (e.g. Google glass). In addition, because of high frequency using, this kind of close relationship makes wearable devices be more personalized than normal mobile devices.

However, the most important problem that manufacturers must consider carefully is the effect brought by wearable devices to the user due to the way of wearing such as the comfortableness and the influence on user’s daily life [10], for example, some user who do not wear glasses may feel uncomfortable while wearing Google glass.

2.5 **Accelerometer**

Acceleration is the rate of the change of velocity which is used for describing the motion of an object physically. Accelerometer is the device that measures the proper acceleration. In this project, acceleration is one of the most essential parameters to determine the motion of the device. According to Newton’s Second Law, the acceleration of a single object in a single direction can be obtained by the formula:

$$F = ma$$

Where F is the resultant force acting on the object, m is the mass, and a is the acceleration. Based on this theory and equation, acceleration can be calculated by measuring the resultant force and the mass. According to the definition of acceleration, another way to obtain acceleration is:

$$v = \frac{\mathrm{ds}}{\mathrm{dt}} \qquad \mathrm{a} = \frac{\mathrm{dv}}{\mathrm{dt}}$$

velocity (v) and distance (s) can be obtained theoretically if the time duration is given.

Traditional accelerometer works in a mechanical way. Conceptually, a mechanical accelerometer can be said as an object with known mass connected with a spring. When the accelerometer experiences an acceleration, the spring will be stretched or compressed depending on the direction of the acceleration. And the resultant force can be measured by the displacement of the spring.

Modern accelerometers are usually tiny MEMS (micro electro-mechanical systems) chips which are made up of components between 1 and 100 micrometers in size. Unlike traditional mechanical accelerometers, there are mainly two types of accelerometers: Capacitive accelerometer and Piezoelectric accelerometer. With similar structures, the main difference between these two types of accelerometers is the way to measure the force and transform it to a measurable and quantifiable value:

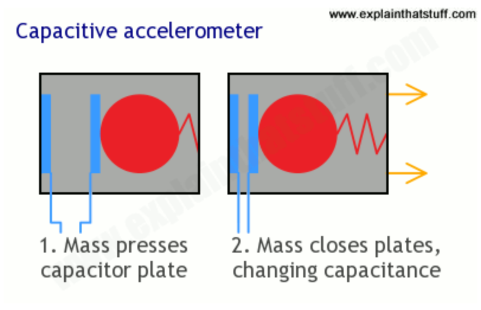

Capacitive accelerometer
www.explainthatstuff.com
1. Mass presses capacitor plate
2. Mass closes plates, changing capacitance

Figure 4 Capacitive accelerometer [11]

Figure 4 shows a simple structure of a capacitive accelerometer, and briefly explains how does a capacitive accelerometer work. The accelerometer contains an object and a capacitor whose plate can be moved to detect the force by sensing the change of capacitance brought by the acceleration. When the sensor is being accelerated, the plate of the capacitor will move which results in the change of the distance between two plates thus the capacitance will have corresponding changes. And the acceleration can be obtained by measuring the value of the capacitance.

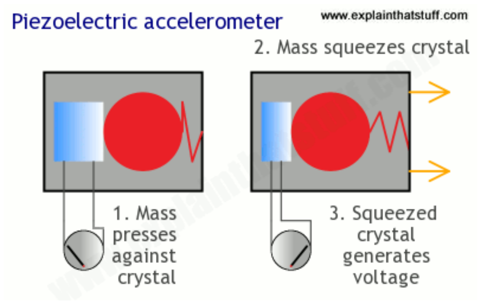

Piezoelectric accelerometer
www.explainthatstuff.com
2. Mass squeezes crystal
1. Mass presses against crystal
3. Squeezed crystal generates voltage

Figure 5 Piezoelectric accelerometer [12]

The other type is called Piezoelectric accelerometer. It uses a piece of piezo-electronic crystal such as quartz to sense the change of force which causes acceleration. When the sensor is being accelerated, the crystal will be pressed or stretched and the voltage generated by the crystal will have a corresponding change. Then the acceleration can be obtained by measuring the voltage. [13]

2.6 **Gyroscope**

The classic gyroscope, also called rotary gyroscope, is an application based on the law of conservation of angular momentum that the total angular momentum of a system is remaining constant in both magnitude and direction if the resultant external torque acting upon the system is zero. This is the reason that models of gyroscope usually contain a spinning disk and several gimbals. When the disk is spinning, the orientation of the gyroscope will maintain constantly, and the angular velocity caused by external torques or rotations applied to the gyroscope can be measured.

However, it is hard to achieve a rotary structure in an electronic device, even fabricate it on a silicon wafer. Vibrating gyroscope was developed as a MEMS (Micro-machined Electro-Mechanical Systems), which are easily available commercially, affordable, and small. [14] This kind of gyroscope are based on relative theory of traditional gyroscope but works in a different manner.

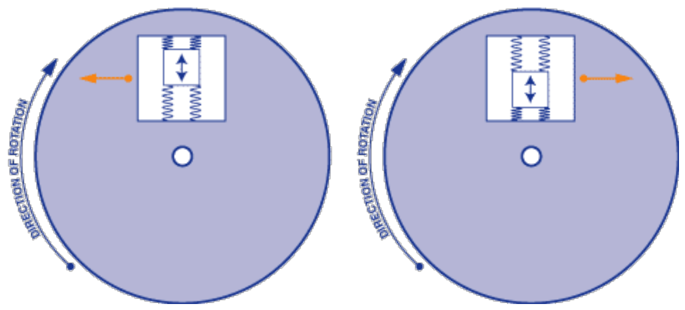


Figure 6 a simplified structure of a mems gyroscope [15]

Vibrating gyroscope is an application of the theory of Coriolis force which is an inertial force. In a rotating system, the rotational velocity of each point is the same. As an object approaches the axis of rotation, the rotational velocity remains the same, but the speed in the direction perpendicular to the axis of rotation will decrease. As shown in Figure 6, when the gyroscope is rotated, a small resonating mass will be shifted as the angular velocity changes. And this movement will be converted into very low-current electrical signals that can be amplified and read by a host microcontroller. [16]

2.7 **Wireless Communication**

Wireless communication, sometime simply called wireless, is a type of transferring information or power between devices without any physical connections (e.g. wires or cables). Based on this definition, broadly speaking, wireless communication systems can be said that was developed at pre-industrial age. These systems were implemented by mean of smoke signals, torch lighting, flashing mirror for examples. However, information transferred within these systems was limited within the line of sight, although the effective distance can be extended by telescopes. [17] Nowadays, "wireless communication" usually represent a series of technologies that transferring continuous signals. The first wireless communication was based on radio transmission which was demonstrated by Guglielmo Marconi in 19th century, from Isle of Wight to tugboat 18 miles away. [18]

After several decades development, wireless communication has become a widely used technology, many applications that are quite common in our daily life are based on wireless communication. For instance, satellite communication (based on modulated microwave), IR communication (based on infrared radiation), FM (frequency modulation) radio, LTE (long-term-evolution, which is not a type of technology, but a standard for high-speed wireless communication for mobile phones and data terminals), Bluetooth (standard for wireless communication), Wi-Fi (WLAN, wireless local area networking based on IEEE 802.11 standards) and so on.

The advantages of wireless communication are obvious. 1). Data is transferred wirelessly which increases the mobility of users. 2). Networks can be accessed easily from anywhere covered by the network regardless of the limitation of the physical cables and ports. 3). The number of users that can access the network will not be limited by physical cables. [19]

III. Technical Specification

3.1 Overview

Last chapter gives a brief introduction about the working manner of traditional mice, which can be summarized as three main steps: 1). Collecting physical motion information from human. 2). Converting the motion information to measurable digital data. 3). Transfer these data to parameters which can be recognized by PC to control the cursor.

In this chapter, technical specifications will be given, especially the hardware and software chosen for the project, including introductions and features of each device and software. And some alternative methods (especially in obtaining the angular data and building the interface between PC and Arduino) will also be discussed in this chapter. To explain matters clearly, some pictures from other sources are used.

3.2 Arduino Uno and Arduino Mini Pro

Arduino Uno is the most common microcontroller board in Arduino product series. It contains nearly all the essential components that a microcontroller should have, for example, 14 digital input/output pins of which 6 pins can be used as PWM outputs, 6 analog input pins, analog-to-digital converter, digital-to-analog converter, a 16 MHz quartz crystal, USB connection and so on. And Arduino Uno also supports SPI and I2C for communicating. It can be said as the best board for entry level developers, also a good choice in experiment for developers. Considering these characteristics, An Arduino Uno is an ideal choice for testing and the first prototype building in this project. [20]

Arduino Pro Mini is an advanced version of the original Arduino board. The most improvement is its size that the PCB is only approximately 0.7" (1.8 cm) x 1.3" (3.3 cm) with essential components. It contains 14 digital input/output pins of which 6 pins can be used as PWM outputs, and supports SPI and I2C as well. In order to reduce the size of the board, USB connection and power jack are removed. And a FTDI board (which can convert data to serial signals) or relative breakout board is needed to build the communication between Arduino Pro Mini and PC and upload the code. [21]

3.3 MPU6050

For this project, an IMU (Inertial Measurement Unit) called MPU6050 was chosen which is a member of the family MPU60X0 developed by InvenSense. The MPU60X0 is the world's first integrated 6-axis motion tracking device that combines a 3-axis gyroscope, 3-axis accelerometer and a Digital Motion Processor™ (DMP) all in a small 4x4x0.9mm package. [22] And it has become one of the most popular integrated sensor in motion measuring e.g. Quadro copter and balancing robot.

The MPU-60X0 features six 16-bit analog-to-digital converters (ADCs) for digitizing the outputs from gyroscope and accelerometer (three axes for each). For precision tracking of both fast and slow motions, the parts feature a user-programmable gyroscope full-scale range of ±250, ±500, ±1000, and ±2000°/sec and a user-programmable accelerometer full-scale range of ±2g, ±4g, ±8g, and ±16g. Communication is performed using I2C (Inter-Integrated Circuit) at 400kHz for MPU6050. [23]

3.4 **Bluetooth**

Bluetooth is a wireless communication technology standard which intends to replace cable connecting potables and fixed electronic devices over short distance. First invented by Ericsson in 1994, Bluetooth now has become a widely-used technology over the world, and the latest version Bluetooth 5 has been unveiled in 2016 by Bluetooth SIG (Special Interest Group) which is a non-profit, non-stock organization with over 20,000 members which mainly deals with setting standards, licensing and advancing Bluetooth capabilities. [24]

Using UHF (Ultra High Frequency) radio wave in the ISM (Industrial, Scientific and Medical) radio band from 2.4 to 2.485 GHz, Bluetooth has a limited transmit distance that the connection will be unstable if the distance exceeds the limit. Due to this characteristic, Bluetooth are mostly used in short-range personal mobile devices communicating, rather than long distance and high-speed data exchange like Wi-Fi. [25] And Bluetooth is defined as three classes according to the transmit power which is also represents the maximum range for the communication: class 1 can reach up to 100m, class 2 can reach 10m, and for class 3, the maximum range is 10cm. [26]

3.5 **Java**

Java is a programming language created by James Gosling at Sun Microsystems in 1991 which was acquired by Oracle Corporation later. The purpose of creating Java was intending to let application developers “write once, run anywhere” (WORA). Since the first version was released in 1995, Java have become one of the most popular programming language in use, and there are 9 million Java developers worldwide reported. [27]

Since Java is a C-language derivative, its syntax rules and format are much similar to C-language. But there are still some differences between these two languages. One of the main difference is the structure of the program. When writing code in C-language, programs always begin from a main function and some other functions following. In Java, the program starts with packages, within the packages are classes, and methods, constants etc. are included in classes. [28]

Java also has some properties as following: [29]

- Platform independent: Java do not access operating system directly, programs use Java Virtual Machine(JVM). In other word, Java programs can run on all supported platforms which makes Java highly portable.
- Object-orientated programming language: programs are based on the concept of “objective”
- Interpreted and compiled language: Unlike normal compiling process, Java source code is transferred into the bytecode. Then these bytecode instructions will be interpreted by the Java Virtual machine (JVM) which translates performance critical bytecode instructions into native code instructions.

3.6 **Processing**

Processing is an open-source programming language created by Ben Fry and Casey Reas in the spring of 2001. The main characteristic of processing is its strength in creating graphs and visual, interactive media, which is one of the idea from the beginning. Processing was designed for users who do not have programming experience or programming starters who can writing code with the aid from visual feedback. Building on Java language and using simplified syntax rules, code in processing is friendly to users which is easier to understand and learn for all users. And the success in programing education made it popular among students, even attracted users from other areas such as art, design and architecture. [30]

Since Arduino IDE is developed based on processing, the structure of programs and syntax rules are quite similar, which makes processing always be the first choice for developers who are working on an Arduino project and planning to build communications between Arduino and PC.

3.7 Data Processing

Since the angles and accelerations (also temperature) measured and converted to digital data by MPU6050 will be stored in 16 bits registers, the raw data read from MPU6050 registers directly will be in the range of [-32767, +32767]. And it is necessary to use some method convert the raw data back to analog value.

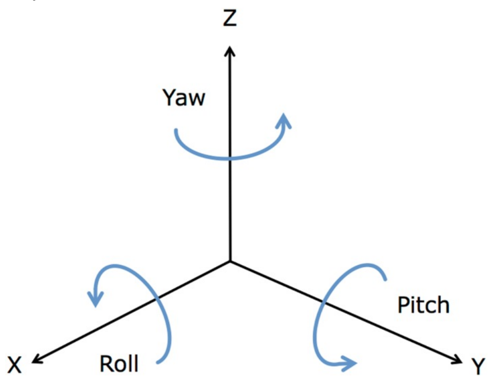
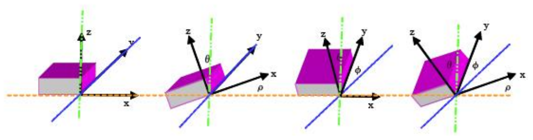


Figure 7 Euler Angles [31]

Euler angles, which are defined as Roll, Pitch and Yaw angle respectively, introduced by Leonhard Euler to describe the orientation of a rigid body in a fixed coordinated system. Any orientation can be said as a product composed by these three element rotations. However, MPU6050 can only measure the accelerations on each axis separately, rather than Euler angles as expected. It is necessary to use some mathematical methods to convert three-axes accelerations to Euler angles. This section will give a brief introduction and explanation about alternative methods in calculating the angles.

3.7.1 Geometrical Calculation



$$\rho = \arctan\left(\frac{A_X}{\sqrt{A_Y^2 + A_Z^2}}\right)$$

$$\phi = \arctan\left(\frac{A_Y}{\sqrt{A_X^2 + A_Z^2}}\right)$$

$$\theta = \arctan\left(\frac{\sqrt{A_X^2 + A_Y^2}}{A_Z}\right)$$

Figure 8 equations for calculating Euler angles [32]

As shown in Figure 8, three Euler angles are usually denoted as φ (sometimes ρ), θ, and ψ. This is the simplest way to calculate the Euler angles. Since the gravity is always affecting the sensor, and this force can be separated into three fractions on each axis. And the orientation can be obtained by measuring the value of each fraction force and reconstructing the gravity.

According to the datasheet of MPU6050, the accelerometer has a scale factor which is depending on the full-scale range set by the program (default range is ±2 g). The acceleration can be simply obtained from the raw data divided by the scale factor. In fact, if using this method, the process of obtaining the real value of accelerations can be omitted due to the ratio calculation. Also, there might be several ways of geometrical calculation.

As for gyroscope, the result returned by the sensor is the angular velocity which means the rate of the angular change. It can be presented as the following equation:

$$\omega = \frac{d\theta}{dt}$$

And angles can be obtained by integrating the angular velocity:

$$\theta = \int_0^t \omega \; dt = \sum_0^t \omega \cdot T$$

Where T is a constant short interval which can be said as approximate dt. By integrating the angles on each axis, Euler angles can be obtained.

However, it is not a good choice to simply use either acceleration or gyroscope to calculate the rotary angles. When using accelerometers, the ideal situation is that the resultant force is only the gravity, where the orientation can be obtained by calculating the fraction of gravity on each axis. But the reality is that the sensor cannot always stay motionlessly, when the sensor is moving, there are a lot more forces driving the sensor rather than pure gravity that the orientation obtained will result in errors even a very small force.

The problem of using gyroscope comes from the integration calculation. This process causes the tendency to drift, which means that the angle may not come back to zero when the sensor is moved back to its original orientation. And there is no approach to offset these errors. In other words, errors no matter it is big or small, will be accumulated over time due to the integration and become a huge error. Therefore, it is not accurate enough when simply only use either accelerometer or gyroscope to calculate the rotary angles.

3.7.2 Digital Motion Processor and Quaternion

There is an embedded module called Digital Motion Processor (DMP) in MPU6050, which is developed by InvenSense. The purpose of the DMP is to offload the pressure of both timing requirements and processing power from the host processor. Typically, motion processing algorithm should be run at a high frequency around 200Hz, to guarantee the precision that all the rotation will be recorded without loss. However, sometimes application may update at a much lower level which cannot match the frequency of the motion processor. In this situation, DMP can simplify timing and power, even the software architecture. [33] The raw results output by DMP are in the form of quaternion. And it is necessary to convert quaternion to Euler angles.

The quaternion is a kind of noncommutative division algebra first invented by Irish mathematician William Rowan Hamilton in 1843 and applied to mechanics in three-dimensional space. In describing positions in three-dimensional space, quaternion has better performance than using Euler angles. For instance, quaternion can avoid a common problem in using Euler angles called gimbal lock which means the object will lose a degree of freedom when two of three gimbals become parallel. A quaternion is generally represented in the following form:

$$\mathrm{H} = \mathrm{a} + \mathrm{b}\boldsymbol{i} + \mathrm{c}\boldsymbol{j} + \mathrm{d}\boldsymbol{k}$$

Where a, b, c, d are real numbers and $\boldsymbol{i}, \boldsymbol{j}$ and $\boldsymbol{k}$ have the properties:

$$\boldsymbol{i^2 = j^2 = k^2 = ijk = -1}$$

$$\boldsymbol{i \times j = k \quad j \times k = i \quad i \times k = -j}$$

From the properties above, it is clearly that quaternion can be said as an extend of complex numbers, since they have similar form and properties. And quaternion can also be represented in the form of vector:

$$\mathbf{q} = [\mathbf{q2} \quad \mathbf{q2} \quad \mathbf{q3} \quad \mathbf{q4}]^{\mathbf{T}}$$

Just like Euler's formulas which establish the relationship between the trigonometric functions and the complex exponential function. Quaternion also has a corresponding relationship with Euler angles:

$$\begin{bmatrix} \phi \\ \theta \\ \psi \end{bmatrix} = \begin{bmatrix} \operatorname{atan2}\left\{2(q_1q_4 + q_2q_3), \left[1 - 2(q_1^2 + q_2^2)\right]\right\} \\ \sin^{-1}\left[2(q_2q_4 - q_1q_3)\right] \\ \operatorname{atan2}\left\{2(q_3q_4 + q_1q_2), \left[1 - 2(q_2^2 + q_3^2)\right]\right\} \end{bmatrix}$$

Figure 9 Euler angles derived from quaternion [34]

And quaternions can be updated by the equation:

$$\begin{bmatrix} \dot{q}_1 \\ \dot{q}_2 \\ \dot{q}_3 \\ \dot{q}_4 \end{bmatrix} = \frac{1}{2} \begin{bmatrix} 0 & \omega_z & -\omega_y & \omega_x \\ -\omega_z & 0 & \omega_x & \omega_y \\ \omega_y & -\omega_x & 0 & \omega_z \\ -\omega_x & -\omega_y & -\omega_z & 0 \end{bmatrix}_B \begin{bmatrix} q_1 \\ q_2 \\ q_3 \\ q_4 \end{bmatrix}$$

Figure 10 equation for updating quaternion [35]

Where $\omega_{x,}$ ω_y and ω_z are the angular velocity on corresponding axis. Using the equations shown in Figure 9 and Figure 10, real time measuring in the form of the Euler angle can be achieved.

3.7.3 Kalman Filter

According to the disadvantages of using accelerometers and gyroscope separately, the performances of accelerometer and gyroscope are good during different period: accelerometer is more accurate in long term, when the object is static, in other words, the sensor is not effected by other forces except gravity. And gyroscope works well in short term which can minimize the accumulation of small errors. Therefore, it becomes a reasonable idea that merging the results both from different periods where each sensor has good behavior and using some mathematical methods to obtain the accurate value of rotary angles. And Kalman filter is one of this kind of method.

Kalman filter was developed around 1960, Named after Rudolf E. Kálmán who was one of the primary developer of the theory. Because of its satisfying performance in estimating from noises with less calculation requirement, it has numerous usage in engineering, especially in global positioning system, aircraft, spacecraft even in guidance and navigation systems. It is also called the best algorithm so far for calculating the posture of an object in three-dimensional space.

Basically, the result of Kalman filter is a combination of the estimates from the previous states and the observation of the state (for a hidden system) at present plus the errors. Before explaining the theory, some definitions are needed to be mentioned:

- $\hat{x}_{k-1|k-1}$ called previous state, which means the state at time k-1 based on the previous estimated states before.
- $\hat{x}_{k|k-1}$ called priori state, which means the estimated state at current time k based on the previous state at k-1 and estimated states before.
- $\hat{x}_{k|k}$ called posteriori state, which means the estimated state at current time k based on the observation up to and including time k.

And the state of the system at time k can be presented as:

$$x_k = \mathbf{F}x_{k-1} + \mathbf{B}\dot{\theta} + \omega_k$$

Where x_k and x_{k-1} are states at time k and k-1. $\dot{\theta}$ is the angular velocity. **F** and **B** are defined as following:

$$\mathbf{F} = \begin{bmatrix} 1 & -\Delta t \\ 0 & 1 \end{bmatrix} \qquad \mathbf{B} = \begin{bmatrix} \Delta t \\ 0 \end{bmatrix}$$

And ω_k is called process noise which is Gaussian distribution with zero mean and covariance Q_k at time k. According to the equation above, the priori state prediction can be obtained by:

$$x_{k|k-1} = \mathbf{F}\hat{x}_{k-1|k-1} + \mathbf{B}\dot{\theta}$$

Let P be the covariance of the priori estimates, and this covariance is also based on the previous covariance given by:

$$P_{k|k-1} = \mathbf{F}\, P_{k-1|k-1}\, \mathbf{F}^{T} + \mathbf{Q}_k$$

$P_{k|k-1}$ is used to calculate Kalman gain K_k which is:

$$K_k = P_{k|k-1}\, \mathbf{H}^{T} S_k^{-1}$$

The meaning of Kalman gain is how much can the innovation be trusted. The definition of innovation will be explained shortly. And S is called the covariance of the innovation which is:

$$S_k = \mathbf{H}\mathbf{P}_{k|k-1}\mathbf{H}^{T} + \mathbf{R}$$

The innovation, symbolled as $\tilde{y}_k$, is the difference between the measurement z_k and the priori estimate $x_{k|k-1}$, which is given by:

$$\tilde{y}_k = z_k - \boldsymbol{H}\hat{x}_{k|k-1}$$

The observation at time k, also written as z_k is obtained from the equation:

$$z_k = \mathbf{H}x_k + v_k$$

Since it is a hidden system that the true state is unknown, all the performance can only be known by observations. **H** is called observation model which is used to map the space of true state into the observation space. And v_k is the measurement noise which is Gaussian distribution as well, with zero mean and covariance R. Because this noise comes from measurement, therefore the covariance R equals to the measurement covariance.

Now the estimate of the state at current time can be predicted as:

$$\hat{x}_{k|k} = \hat{x}_{k|k-1} + K_k\tilde{y}_k$$

This equation shows clearly that the estimate of current time is the product of two parts: 1). the estimate based on the previous time 2). the difference between the measurement and the priori estimate state, with a coefficient Kalman gain. It is more accurate than using measurement or estimate purely since most of the error can be filtered. [36]

Although Kalman filter can provide a good performance in obtaining the state of a hidden system, it still has some disadvantages. The main problem is that the requirement of the noise which needs to be Gaussian distribution but in fact the noise cannot behave as the distribution strictly. Another problem is that Kalman filter requires higher mathematical ability to comprehend and it is more complex to be achieved by programming, especially for those platform that does not support linear algebra and vector calculation. For testing, there is an opensource Kalman filter on GitHub, but the performance is not so satisfying that it needs several seconds to obtain a stable value of angles which cannot meet the requirement for a mouse.

3.7.4 Complementary Filter

Since the Kalman filter is more difficult to be implemented, another method called complementary filter was developed which also combines the characteristics from both accelerometer and gyroscope. The mathematical expression of complementary is much more intuitive and simpler than Kalman filter which is:

$$\text{Angle} = \alpha \times (\text{Angle} + \theta \times \text{dt}) + (1 - \alpha) \times a$$

Where Angle is the rotary angle on the specific axis, α is a constant, θ is the data from gyroscope which is the angular velocity, and a is the data from the accelerometer. The theory of complementary filters is not as difficult as Kalman filters'. The main point is applying a low pass filter to the gyroscope which can filter the long-term result to minimize the influence brought by the potential of drift. As for accelerometer, a high pass filter is applied, and noise of accelerometer in short term will be filtered. The final angle is the product of short term result from gyroscope and long-term result from gyroscope and each part will have good performance due to the characteristic of two sensors. [37]

Focusing on the equation, it is more intuitive to explain the theory of the filter. When the sensor is rotated, the data from the gyroscope takes the main part (since α is much larger than (1 - α)), and the rotary angle nearly equals to the angle at previous time plus the integration of angles obtained by gyroscope which is the integration of the angular velocity. When the sensor is motionless which can be said as static state, and there is no angular velocity on any axis, θ will be zero in this situation, therefore the angle will converge to the previous value of angle plus the angle obtained from the accelerometer.

As for the constant α means how much the result can be trusted. It is not the same for different devices but α is between (0, 1) and usually larger than (1 - α) in order to guarantee the precision of the gyroscope. Some tutorial used β to represent (1 - α) as another parameter and clarify the constraint that the sum of α and β is 1. This is because the complementary filter is the combination of two parts, and this combination filter should cover the same range as the original signal, in other words, the complementary filter is an all-pass filter.

IV. Methodologies

4.1 **Overview**

From the specification of last chapter, a working manner of the air mouse in this project can be clarified as shown in Figure 11 below:

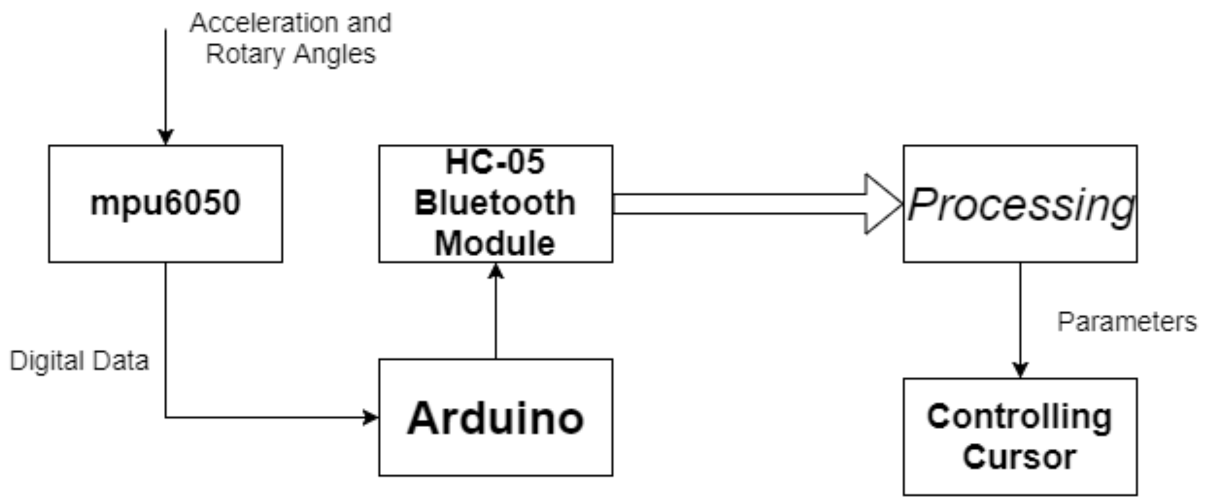


Figure 11 working manner of the air mouse

After measured by mu6050, acceleration and rotary angles in the form of digital data will be collected by Arduino. These data will be sent to PC as serial signals by HC-05 Bluetooth module. And *Processing* will project these rotary angles to parameters which then can be used to control the motion of the cursor.

This chapter will mainly explain the methodologies used in this project which are mostly programming using Arduino IDE and some hands-on work, such as soldering to build the connections between components, sensors and Arduino. According to the aims and the technical specifications, all the work can be separated as several parts. Each part will explain the components or devices chosen for the project, testing work, and some details during the development.

4.2 **GY521**

GY521 is a kind of breakout board using MPU6050 which is chosen to use in this project. It's not clear who developed this module first, but it is popular because of its low-cost and easy-to-use. Different from other kinds of breakout board, GY521 has a voltage regulator which allows the board to work with 5V. If using 3.3V as voltage supply, it may influence the performance of I2C. Fortunately, the influence is slight and the sensor can work under 3.3V according to the experiment.

GY521 has 8 pins which are: Vcc and GND are for power supplying, SCL and SDA are for I2C communication. Unlike other common breakout boards supporting I2C, GY521 contains another two pins, XCL and XD which are used for sub-I2C to be a master controlling other boards, in other words, GY521 can be both master and slave, for example, it can be a slave when communicating with Arduino but be a master controlling a magnetometer, but these two pins will not be used in this project. AD0 which is used to initialize the address of the chip, and INT which will not be used in this project. [38]

Before writing the program, testing work is necessary. Some open-source code will be used based on the online tutorials. Testing includes two steps: First, using open-source code to make sure the chip is working and raw data can be read by Arduino through I2C Bus and then transmitted to PC. Second, translating the raw data into an acceptable format and using the serial monitor from Arduino IDE to display the data for the further steps.

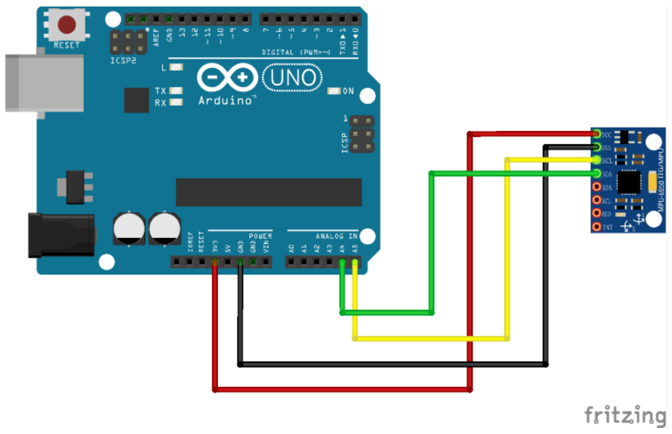


Figure 12 connection between Arduino and GY521

As shown in Figure 12, the wiring for the test is simple: Vcc and GND connect to the corresponding ports on the Arduino, SCL is connected to port A4, and SDA connects to A5. Since it is just for testing the sensor, Bluetooth module was not used, data read by GY521 was sent through the USB cable and displayed by the built-in serial port monitor from Arduino IDE.

```
// MPU-6050 Short Example Sketch
// By Arduino User JohnChi
// August 17, 2014
// Public Domain
#include<Wire.h>
const int MPU_addr = 0x68; // I2C address of the MPU-6050
int16_t AcX, AcY, AcZ, Tmp, GyX, GyY, GyZ;
void setup() {
   Wire.begin();
   Wire.beginTransmission(MPU_addr);
   Wire.write(0x6B);   // PWR_MGMT_1 register
   Wire.write(0);      // set to zero (wakes up the MPU-6050)
   Wire.endTransmission(true);
   Serial.begin(9600);
}
```

```
void loop() {
  Wire.beginTransmission(MPU_addr);
  Wire.write(0x3B);   // starting with register 0x3B (ACCEL_XOUT_H)
  Wire.endTransmission(false);
  Wire.requestFrom(MPU_addr, 14, true); // request a total of 14 registers
  AcX = Wire.read() << 8 | Wire.read(); // 0x3B (ACCEL_XOUT_H) & 0x3C (ACCEL_XOUT_L)
  AcY = Wire.read() << 8 | Wire.read(); // 0x3D (ACCEL_YOUT_H) & 0x3E (ACCEL_YOUT_L)
  AcZ = Wire.read() << 8 | Wire.read(); // 0x3F (ACCEL_ZOUT_H) & 0x40 (ACCEL_ZOUT_L)
  Tmp = Wire.read() << 8 | Wire.read(); // 0x41 (TEMP_OUT_H) & 0x42 (TEMP_OUT_L)
  GyX = Wire.read() << 8 | Wire.read(); // 0x43 (GYRO_XOUT_H) & 0x44 (GYRO_XOUT_L)
  GyY = Wire.read() << 8 | Wire.read(); // 0x45 (GYRO_YOUT_H) & 0x46 (GYRO_YOUT_L)
  GyZ = Wire.read() << 8 | Wire.read(); // 0x47 (GYRO_ZOUT_H) & 0x48 (GYRO_ZOUT_L)
  Serial.print("AcX = "); Serial.print(AcX);
  Serial.print(" | AcY = "); Serial.print(AcY);
  Serial.print(" | AcZ = "); Serial.print(AcZ);

  Serial.print(" | GyX = "); Serial.print(GyX);
  Serial.print(" | GyY = "); Serial.print(GyY);
  Serial.print(" | GyZ = "); Serial.println(GyZ);
  delay(1000);
}
```

Figure 13 open source code for GY521 [39]

Figure 13 is the test code used for GY521 which is available at Arduino tutorial. Since the port AD0 is not connected to any pin which means that the default address of this chip is 0×68 on I2C Bus, which should be called when setup the I2C communication. And according to the register map, the register at address 0×6B is PWR_MGMT_1 which is used to control the working state of the sensor. Therefore, in the setup section, a "0" is written to the register to wake the sensor up.

All the data measured by Gy521 will be converted to 16-bit data and stored into the corresponding registers. Therefore, the variable used to store the reconstructed value are defined as 16-bit integers. From the register map, 14 registers whose first address starts from 0×3B are used to store the data. And the function requestFrom() is to send a requirement to a specific address on I2C Bus with the number of bytes requested by the master. Since each data is divided to two parts: named by high and low respectively, and each time read() function can only get one byte which is 8-bit of data. The original value can be reconstructed by shifting the value in high eight bits and having "or" calculation with low eight bits which is equivalent to calculation "plus". The test work of Bluetooth module had not been finished at this stage, all the value will be sent back to PC simply through USB cable, and printed out by serial port monitor.

Another thing needs to be mentioned is that the angles measured by MPU6050 is relative to the position that the chip is parallel to the horizontal plane. A function called Calibration() in the final code is used to record the angles before at the very first so that the sensor can measure the relative angles with respect to the initial posture. Calibration() function will record first 100 angles and calculate the mean value in order to avoid the influence brought by the noise or other unexpected circumstance which may cause imprecisions.

4.3 **HC-05 and BlueSMiRF Silver**

HC-05 and BlueSMiRF Silver are both easy-to-use wireless module breakout board using Bluetooth to communicate with other devices. Both of them can be used for this project and MPU6050 was chosen finally. Introductions and discussions will be given in this section. The graph used to clarify the connection are drawn myself, watermarks are added automatically by the software.

HC-05 is a simple module developed long time ago, and it is not clear who developed this module first. It is compatible with Bluetooth SPP (Serial Port Protocol) profile which is a subset of Bluetooth profile by which the communication between two devices through Bluetooth can be specified. Based on RFCOMM (Radio Frequency Communication) protocol, SPP emulates a virtual cable supporting serial signals which can take the place of RS-232 serial ports. In other words, HC-05 transmits data as serial signals through Bluetooth. [40]

HC-05 contains 6 pins: Vcc and GND for power supplying, TXD and RXD for data transmitting, KEY forces AT Command Setup Mode If brought HIGH before power is applied, and STATE tells if the device is connected or not. And the last two pins will not be used in this project.

BlueSMiRF is a Bluetooth module developed by Sparkfun which is an electronic industry manufacturing and retailing components and breakout boards. Different from the gold version, BlueSMiRF silver is a class 2 Bluetooth module whose transmit range is limited to about 10 meters but with lower power consuming. There is also a 3.3V regulator on the board which allows the board to work under the input voltage between 3.3V to 6V. [41]

Comparing HC-05 and BlueSMiRF, one of the main difference is that BlueSMiRF module supports both SPP and HID firmware which will make the process of developing the driver program easier. However, HC-05 and BlueSMiRF uses the same main chip (BC417 produced by Cambridge Silicon Radio) inside the breakout board, what makes the difference is the firmware it is running. Because of this advanced firmware, the price of BlueSMiRF module is much higher which cost about ￡30 than HC-05 which costs only about ￡5. Based on the object of this project, HC-05 was chosen finally.

There are two solutions which can solve this problem: the first one is remove the firmware inside the HC-05 and flashing the firmware copied from BlueSMiRF, then HC-05 is able to support HID. But this solution cost more time in changing the firmware and also it is kind of infringement. The second solution is not using HID for building the interface and find other ways which are based on serial signals. And this problem will be explained in sections behind.

For the test of HC-05, an open-source code was used which is to control a LED by sending instructions to the Arduino, and a respond message will be sent back to PC through Bluetooth.

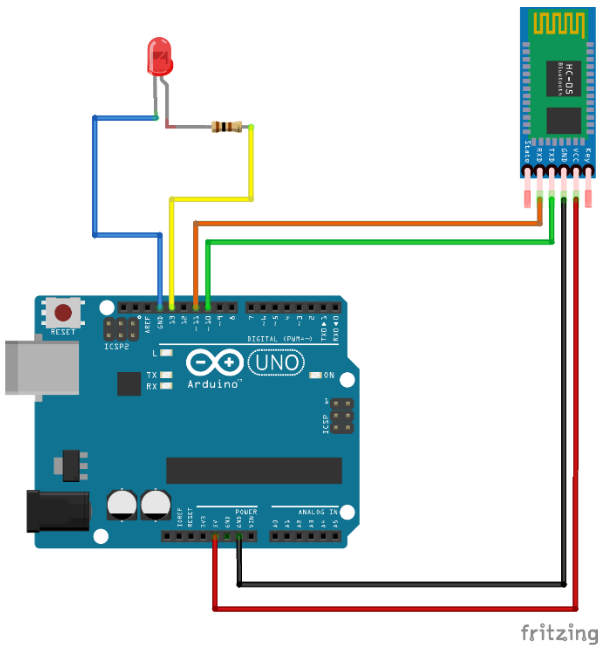


Figure 14 connection of testing HC-05

As shown in Figure 14, TXD and RXD are connected to the pin 10 and 11 respectively according to the opensource code which defines the port 10 for transmitting data and port 11 for receiving data. And a LED with a 100Ω resistor are connected to the port 13 and GND which is used for checking if the Arduino received the instruction from PC. In order to guarantee that data was transmitted and received by Bluetooth module, USB was removed after the program was uploaded to Arduino, and a mobile power bank was used for power supplying.

```
// This program shown how to control arduino from PC Via Bluetooth
// Connect ...
// arduino>>bluetooth
// D11     >>>   Rx
// D10     >>>   Tx
//Written By Mohannad Rawashdeh
//for http://www.genotronex.com/
// you will need arduino 1.0.1 or higher to run this sketch

#include <SoftwareSerial.h>// import the serial library
SoftwareSerial Genotronex(10, 11); // RX, TX
int ledpin = 13; // led on D13 will show blink on / off
int BluetoothData; // the data given from Computer
```

```
void setup() {
  Genotronex.begin(9600);
  Genotronex.println("Bluetooth On please press 1 or 0 blink LED ..");
  pinMode(ledpin, OUTPUT);
}

void loop() {
  if (Genotronex.available()) {
    BluetoothData = Genotronex.read();
    if (BluetoothData == '1') { // if number 1 pressed ....
      digitalWrite(ledpin, 1);
      Genotronex.println("LED   On D13 ON ! ");
    }
    if (BluetoothData == '0') { // if number 0 pressed ....
      digitalWrite(ledpin, 0);
      Genotronex.println("LED   On D13 Off ! ");
    }
  }
  delay(100);// prepare for next data ...
}
```

Figure 15 open source test code for HC-05 [42]

There are two small LEDs on HC-05 chip which are used to show the state of the device. When the HC-05 is vacant, in other words, not pairing with other devices, LEDs will blink continuously. When pairing with HC05 is successful, the lights will blink twice each time with a short delay. And PC will receive a message "Bluetooth On please press 1 or 0 blink LED ..." from Arduino due to the program.

Then the LED connected to the Arduino can be controlled by sending “0” or “1” to the Arduino through the serial port monitor. According to the code, when receiving “0”, the voltage of port 13 will be low, the LED will be off, and it will be turned on when receiving “1”. Each time an instruction is sent, Arduino will send back a message about the current state of the LED ("LED   On D13 ON/Off ! ").

4.4 Interface Building

From the research, there are three methods that can achieve the interface including cursor controlling and clicking functions: HID (Human Interface Device), Arduino built-in libraries called mouse.h and a class built in Java called Robot class. Brief introductions and some discussions about these three methods will be given in this section.

Human Interface Devices (HID) are a kind of method that provide an interface between human and electronic information system, this interface can be either input (e.g. keyboard and mice) or output (e.g. speaker and headset).

For keyboard, mice and other kinds of controllers, HID are also called HID class, as a part of the USB specification computer peripherals. For creating an interface between USB device and PC, if the interface does not fit in one of the pre-defined classes, developers will have to develop a custom driver program to define all the relative information. HID-class devices simplify USB communication one step further by using a standardized, flexible driver that comes pre-installed with all commonly-used operating systems, which allows developers to create communications without custom driver development. [43]

For this project, the main problem developing a custom HID driver is the firmware of the Bluetooth module. Since HC-05 was chosen, the firmware has to be changed to support HID which will cost more time in researching and experiments.

<mouse.h> is a set of libraries that enable Arduino to control the cursor and keyboards, in other words, using these libraries, Arduino will be able to behave as a combination of mouse and keyboards. The main advantage of this method is that it is easier to develop the device especially in programing since. It does not need to write code for PC to receive data and move the cursor, all the functions can be done only using Arduino IDE which will save plenty of time. [44]

Unfortunately, these libraries only support a limited range of types of Arduino which does not include basic series like Arduino Uno and Pro Mini. If using this method, the type of Arduino must be changed and the size of the project cannot be minimized.

Classes are a one of the very basic concepts in objective oriented programming. Each class is like a blueprint or template to construct a specific object. In Java, a class includes details to approach the objective, e.g. data type and methods. [45]

Robot class is a built-in class which is used to generate native system input events for testing automation, self-running demos, and other applications where control of the mouse and keyboard is needed. Unlike posting events to the AWT event queue or AWT components which just generates input events, Robot class will have corresponding reaction in reality according to the instructions e.g. moving the cursor and key pressing. Since *Processing* is built based on Java, Robot class has been included since early version of *Processing*.

The advantage of using this method is that custom driver is not needed any more, the data in the form of rotary angles can be used directly since the parameters of the function mouseMove() which is used to control the cursor in Robot class are the coordinates within the operating system. Also, Arduino IDE is developed based on *Processing*, the compatibility between PC and Arduino would be better. Considering these circumstances, the method using *Processing* was chosen eventually.

There is no open source code just for testing this function, and the final code was finished by reading relative tutorials and experiments. Some problem occurred during this section will be mentioned and explained in the next section.

4.5 **Algorithm Implementation**

According to the aims and the technical specifications, the project can be separated as several steps:

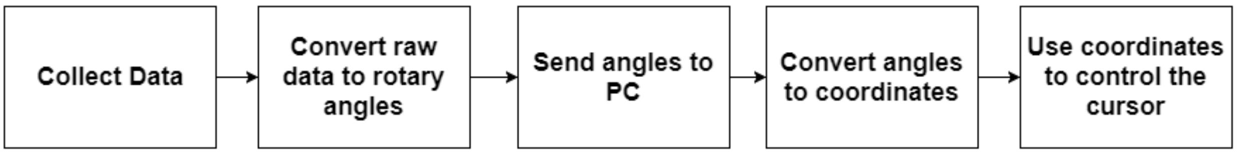

Collect Data
Convert raw data to rotary angles
Send angles to PC
Convert angles to coordinates
Use coordinates to control the cursor

Figure 16 Flow chart of the project

Form previous sections, the problem of collecting data, converting raw data to rotary angles, sending data through Bluetooth and controlling the cursor have been solved. The remaining parts will be explained in this section.

Since data is transmitted through HC-05 which only supports communications in the form of serial signals, a built-in library called SoftwareSerial is necessary which can change Arduino board pin to serial pin. Within the SoftwareSerial library, there are two functions that can be used to send data which are println() and write(). The main difference is that println() will send the data as a character string and write() will only transmit data as a raw byte. Considering the angles needed to be transmitted belong to different categories (roll angle and pitch angle), some flag characters are needed to be transmitted as well so that different angles can be distinguished, println() was chosen finally.

A problem occurred because of println() function is that when sending the data as a string, println() function will add a carriage return character (ASCII 13, or '\r') and a newline character (ASCII 10, or '\n') automatically. They will not be printed out but the character string cannot be transferred to float variables directly for further calculations. To solve this problem, two methods have been tested. The first method is separating the character string to several single characters and then reconstructing all the characters without other Unicode "nbsp" (non-breaking space) characters. The second method is much easier, there is a function called trim() which can remove whitespace characters from the beginning and end of a string, even “nbsp” characters.

Since MPU6050 does not contain a magnetometer which means that MPU6050 cannot obtain the absolute value of yaw angle. Also, the yaw angles obtained by the methods above (section 3.7) are not satisfying, the value keeps increasing when the sensor is static. Based on this reason, yaw angle was not considered as a parameter for projecting angles to coordinates. It is still necessary to distinguish which angle the value received belongs to. The algorithm implemented is sending a flag character each time before the angle value is sent, as shown in Figure 17. And the category of each angle can be known by checking the flag character using *Processing*. Since the flag character is also sent as a character string following by two more characters, but only the first one character is needed, function charAt() can solve this problem which can take the character at a specific position within a string.

Sometimes it may happen that Processing will receive nothing which will be displayed as null. Usually this situation occurs when the frequency of sending the data by Arduino and the frequency of printing the data by *Processing* are not matched. For this circumstance, each time before printing out the data, checking is essential to guarantee that data is truly received.

```
if(val != null)
{
     first = val.charAt(0);
     if (first == 'R')
   {
       Roll = myPort.readStringUntil('\n');
       if(Roll != null)
       {roll = float(trim(Roll));}
       /*println("roll:"); //print it out in the console
       println(roll);*/
   }
       else if (first == 'P')
   {
       Pitch = myPort.readStringUntil('\n');
       if(Pitch != null)
       {pitch = float(trim(Pitch));}
       /*println("pitch:");
       println(pitch);*/
   }
```

Figure 17 Code for distinguish angles

The algorithm for controlling the cursor is quite simple. Before projecting angles, a “window” function is implemented since the range that human index finger can reach is limited, which is set between $\pm 45°$ according to experiments as the initial angle is set to 0°. The coordinate range of laptop is [0, 1919] on x axis and [0,1079] on y axis obtained by program, and the cursor is expected to reach anywhere within the operating area, therefore the angles are projected simply by mapping the angle value to the coordinate value.

V. Conclusions and Improvements

5.1 **Objective Achievement**

From section 1.2, the objective of this project is to find the possibility of developing a low-cost air mouse prototype based on Arduino. And a list of aims was defined in section 1.3 which separated the whole project to several specific steps. Starting with introductions of background materials, this report gives introductions and discussions about the most popular methods which can be used for achieving each single aim. And details about the implementation and some problem occurred during development are explained in Chapter 4. From the result of test and a simple prototype which used Arduino Uno rather than Pro Mini, the cursor can be moved according to the roll angle and pitch angle, and the time of reaction is acceptable, sometimes it will have a very short latency.

This project so far just achieves the main part of the objective and it still have a long way to become a completed product. As for the cost, from my purchase history, a ideal prototype including a Arduino Pro Mini (£ 1.99 each), a mp6050 sensor (£ 2.7 each), a HC-05 Bluetooth module (£ 6.39 each), a 150mAh Lithium polymer battery (£ 2.2 each) used for power supplying, 2 capacitive touch sensors (£ 1.29 each), costs £ 15.86. Even though including the cost of a FTDI chip which is used for uploading code to Arduino Pro Mini, the total cost is £ 30.25. Considering these are only core components, some other stuff e.g. shell and some fiber are needed to make it become wearable, the cost of completed prototype would be higher, but still less than the price of similar products being sold online. *Mycestro*, a commercial air mouse, which is also the model of this project, costs $149 (now it is on sale that costs $129) on its official website. [46]

5.2 **Improvements**

Although controlling the cursor by the sensor MPU6050 is achieved, the performance is not very satisfying so far. And some further work need to be done to improve. This section will describe some problem have been found so far, and some possible solution which has not been tested will be explained. *Mycestro* and *Myo armband* will be models and some alternative solution will be speculated and discussed based on its official website. Since *Mycestro* is closer to this project, discussions will mainly focus on it.

5.2.1 COM Port Detection

A problem which still needs to be solved is the COM port detecting. After the connection between PC and the Bluetooth module is built, two COM ports will be taken to transmit and receive data, the number of these occupied ports can be checked in Device Manager. And this number are required in the program to build data communication. However, in different situations, the number of ports taken will be different depending on the number of connections between this PC and other devices. It would be better if the program can detect the port which is occupied by the Bluetooth module automatically rather than check the port number manually.

According to *Mycestro* website, it is paired with a USB dongle, and has its own control app which can be used to customize the operations e.g. the sensitivity of the cursor, either vertically (up-and-down) or horizontally (left-and-right), and function of each button. It seems the interface is built by HID protocol.

5.2.2 “Shaking” Problem

Another problem which is partly solved but still need improvement is that the cursor is “shaking” without moving the MPU6050. The reasons cause this problem coming from two parts: 1). There will be some tiny changes in angle since it cannot be held absolute motionlessly by human hand. 2). Although a filter is implemented, there is still some tiny noise from the sensor. And these tiny changes will be amplified when projecting the angle to the coordinate. To solve this problem, two filtering functions have been added to the program both in both Arduino IDE and *Processing* which will simply filter the changes in angles or coordinates within a specific value. As for the result, most of the tiny shaking can be filtered, but there is still some shaking since the change is out of the filtering range. And the motion of cursor becomes a bit rough since the changes when the sensor is truly moved are also filtered.

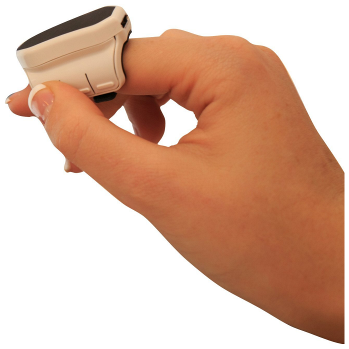


Figure 18 picture of Mycestro [47]

Mycestro provides a good idea: by introducing a stand-by mode, the cursor will only be moved when the thumb (as shown in Figure 18) is touching the sensing area(thumbpad). In other cases that the sensing area is not being touched, by speculating, the device may not send data to PC which can also save power. This idea can avoid the shaking problem when the device is static. And it also solves the conflict that users do not need to move the cursor while they are typing.

However, although the idea is not difficult to be achieved. The problem is the sensor. It seems that it needs two layers of sensors: a capacitive sensor to achieve the stand-by mode on the top, and buttons for clicking functions beneath the capacitive sensor. Since the breakout board of capacitive sensor cannot be such thin and tiny, and in order to achieve stand-by and clicking functions even without scrolling, at least three sensors are needed unless making the operation more complicated, or these sensors will take too much room and will influence the comfortability of fingers.

Another thing needs to be mentioned is that, from the troubleshooting page of the *Mycestro* website, the cursor sometimes will drift without moving the *Mycestro*, and it needs to be recalibrated manually. Since the code and algorithm are inaccessible, speculations are not reasonable enough. But a conclusion can be made which is, *Mycestro* is completed but still not a well-development product.

5.2.3 Algorithm Improvements

The algorithm of moving the cursor is still needed to be improved. The algorithm used so far is just simply getting the angle, and mapped the angle to the range of angle that index finger can move. And the mapping relationship between angles and coordinates is quite simple that pitch angles are mapped to x axis and roll angles are mapped to y axis. However, the motion of human finger is a complex combination of rotation which cannot be described simply by two axes. The problem is that the cursor cannot reach some critical positions e.g. four corners of the operating area, especially the bottom-right corner, which is hard to reach for finger under this algorithm. An alternative solution is introducing yaw angle to the algorithm which will be helpful for describing the posture of the finger. And Kalman filter might be necessary to obtain yaw angle.

Myo armband, another commercial controller (it is not suitable to call it mouse since it has wider usage than air mouse), can achieve controlling by detecting the electrical signals produced by arm muscle activities using EMG (Electromyography) sensors. Although the method is different, but it can be seen from the website, the armband can recognize several basic gestures which seems like classification algorithm or some relative algorithms, and the idea of using machine learning is a good alternative solution to improve the user experience. Users do not need to remember different gestures and corresponding operations which can be customized to meet customer's operating habit as much as possible. For this project, theoretically, it is possible to introduce an algorithm that allows users to define the two axes rather than following the programmed vertical axes, which will let users operate in a more comfortable way.

VI. References

[1] The Computer Mouse: A Timeline
Available at: http://genevalunch.com/wp-content/uploads/2008/12/145378.pdf [Accessed 8 April 2017]

[2] Kickstarter
Available at: https://www.kickstarter.com/discover/advanced?ref=nav_search&term=wearable+controller [Accessed at 8 April 2017]

[3] Computer mice, explainthatstuff
Available at: http://www.explainthatstuff.com/computermouse.html [Accessed 29 July 2017]

[4] Computer mice, explainthatstuff
Available at: http://www.explainthatstuff.com/computermouse.html [Accessed 29 July 2017]

[5] Photo Gallery of - Optical Mouse Vs Laser Mouse, Animalia life
Available at: http://animalia-life.club/other/optical-mouse-vs-laser-mouse.html [Accessed 31 July 2017]

[6] How do optical mice work, howstuffworks
Available at: http://computer.howstuffworks.com/question631.htm [Accessed 31 July 2017]

[7] Arduino Introduction, Arduino
Available at: https://www.arduino.cc/en/Guide/Introduction [Accessed 6 April 2017]

[8] Arduino Introduction, Arduino
Available at: https://www.arduino.cc/en/Guide/Introduction [Accessed 6 April 2017]

[9] Wearable Device, Techopedia
Available at: https://www.techopedia.com/definition/31206/wearable-device [Accessed 6 April 2017]

[10] Wearable Technology: Pros and Cons, humavox
Available at: http://www.humavox.com/blog/wearable-technology-pros-cons/ [Accessed 6 April 2017]

[11] Accelerometers, explainthatstuff
Available at: http://www.explainthatstuff.com/accelerometers.html [Accessed at 9 April 2017]

[12] Accelerometers, explainthatstuff
Available at: http://www.explainthatstuff.com/accelerometers.html [Accessed at 9 April 2017]

[13] Accelerometers, explainthatstuff
Available at: http://www.explainthatstuff.com/accelerometers.html [Accessed at 9 April 2017]

[14] Gyroscope, sensorwiki
Available at: http://sensorwiki.org/doku.php/sensors/gyroscope#fn__4 [Accessed at 9 April 2017]

[15] how a gyro works, Sparkfun
Available at: https://learn.sparkfun.com/tutorials/gyroscope/how-a-gyro-works [Accessed at 9 April 2017]

[16] how a gyro works, Sparkfun
Available at: https://learn.sparkfun.com/tutorials/gyroscope/how-a-gyro-works [Accessed at 9 April 2017]

[17] Goldsmith, A. (2009). Wireless communications. Cambridge [u.a.]: Cambridge University Press. Page:1 [28 July 2017]

[18] Different Types of Wireless Communication with Applications, Tarun Agarwal
Available at: https://www.elprocus.com/types-of-wireless-communication-applications/ [Accessed 28 July 2017]

[19] Different Types of Wireless Communication with Applications, Tarun Agarwal
Available at: https://www.elprocus.com/types-of-wireless-communication-applications/ [Accessed 28 July 2017]

[20] ARDUINO UNO REV3, arduino.cc
Available at: https://store.arduino.cc/arduino-uno-rev3 [Accessed at 2 August 2017]

[21] ARDUINO PRO MINI, arduino.cc
Available at: https://store.arduino.cc/arduino-pro-mini [Accessed at 2 August 2017]

[22] InvenSense Inc., MPU-6000 and MPU-6050 Product Specification Revision 3.4, August 2013, Page: 7 [9 April 2017]

[23] InvenSense Inc., MPU-6000 and MPU-6050 Product Specification Revision 3.4, August 2013, Page: 7 [9 April 2017]

[24] Introduction to Bluetooth Technology, its Working and its Applications, edgefxkits
Available at: http://www.edgefxkits.com/blog/bluetooth-technology-and-its-working [Accessed at 10 April 2017]

[25] What is the difference between Bluetooth and Wi-Fi, Techopedia
Available at: https://www.techopedia.com/2/27881/networks/wireless/what-is-the-difference-between-bluetooth-and-wi-fi [Accessed at 10 April 2017]

[26] Bluetooth Basics, Sparkfun
Available at: https://learn.sparkfun.com/tutorials/bluetooth-basics/how-bluetooth-works [Accessed at 2 August 2017]

[27] JavaOne 2013 Review: Java Takes on the Internet of Things, Timothy Beneke and Tori Wieldt
Available at: http://www.oracle.com/technetwork/articles/java/afterglow2013-2030343.html [Accessed at 31 July 2017]

[28] Java language basics, J Steven Perry
Available at: https://www.ibm.com/developerworks/java/tutorials/j-introtojava1/index.html [Accessed at 31 July 2017]

[29] Introduction to Java programming – Tutorial, Lars Vogel, Simon Scholz
Available at: http://www.vogella.com/tutorials/JavaIntroduction/article.html#introduction-to-java [Accessed 31 July 2017]

[30] Overview, Processing.org
Available at: https://processing.org/overview/ [Accessed at 31 July 2017]

[31] Average roll, pitch, and yaw angles, ResearchGate
Available at: https://www.researchgate.net/figure/262055313_fig2_Average-roll-pitch-and-yaw-angles [Accessed 3 August 2017]

[32] Calculating the roll and pitch of an object, mathhelpforum
Available at: http://mathhelpforum.com/geometry/185200-calculating-roll-pitch-object.html [Accessed at 3 August 2017]

[33] InvenSense Inc., MPU-6000 and MPU-6050 Product Specification Revision 3.4, August 2013, Page: 25 [9 April 2017]

[34] Quaternion,
Available at: http://www.princeton.edu/~stengel/Quaternions.pdf [Accessed at 4 August 2017]

[35] Quaternion,
Available at: http://www.princeton.edu/~stengel/Quaternions.pdf [Accessed at 4 August 2017]

[36] A practical approach to Kalman filter and how to implement it, Kristian Sloth Lauszus
Available at: http://blog.tkjelectronics.dk/2012/09/a-practical-approach-to-kalman-filter-and-how-to-implement-it/ [Accessed at 7 August 2017]

[37] Reading a IMU Without Kalman: The Complementary Filter, Pieter-Jan
Available at: http://www.pieter-jan.com/node/11 [Accessed at 10 August]

[38] MPU-6050 Accelerometer + Gyro, Arduino
Available at: http://playground.arduino.cc/Main/MPU-6050#boards [Accessed at 10 April 2017]

[39] MPU-6050 Accelerometer + Gyro, Arduino
Available at: https://playground.arduino.cc/Main/MPU-6050 #short [Accessed at 10 April 2017]

[40] HC-05 Bluetooth Module Datasheet, [online]
Available at: https://www.gme.cz/data/attachments/dsh.772-148.1.pdf [Accessed at 11 April 2017]

[41] Using the BlueSMiRF, Sparkfun
Available at: https://learn.sparkfun.com/tutorials/using-the-bluesmirf [Accessed 11 April 2017]

[42] Arduino AND Bluetooth HC-05 Connecting Easily, Mohannad Rawashdeh
Available at: http://www.instructables.com/id/Arduino-AND-Bluetooth-HC-05-Connecting-easily/
[Accessed at Accessed at 11 August 2017]

[43] Human Interface Device Tutorial, Silicon Labs
Available at: https://www.silabs.com/documents/public/application-notes/AN249.pdf [Accessed at 10 April 2017]

[44] Mouse and Keyboard libraries, Arduino Reference
Available at: https://www.arduino.cc/en/Reference/MouseKeyboard [Accessed at 10 August 2017]

[45] What Is a Class, oracle.com
Available at: https://docs.oracle.com/javase/tutorial/java/concepts/class.html [Accessed at 11 August 2017]

[46] Mycestro
Available at: http://www.mycestro.com/shop/ [Accessed at 20 August 2017]

[47] Mycestro The Wearable Gesture Based Mouse, Amazon
Available at: https://www.amazon.com/Mycestro-Wearable-Gesture-Based-Mouse/dp/B00U1QYV3K
[Accessed at 18 August 2017]

VII. Appendices

This is the code for processing, some parts are modified based on opensource code. Code within comment are used for debugging.

```
#include <SoftwareSerial.h>// import the serial library
#include <Wire.h>
#include <Math.h>

SoftwareSerial MySerial(10, 11); // RX, TX
int BluetoothData; // the data given from Computer



float R2D = 180/PI;
int MPUAddr = 0x68;
const int nRegs = 7;

int nTimes = 100;
int CalibrationValue[nRegs];

unsigned long lastTime = 0;

float Last_Angle_x = 0.0;
float Last_Angle_y = 0.0;
float Last_Angle_z = 0.0;

float Gyro_x = 0.0;
float Gyro_y = 0.0;
float Gyro_z = 0.0;

float baseAngle_x = 0.0;
float baseAngle_y = 0.0;

float last_outputRoll = 0.0f;
float last_outputPitch = 0.0f;

float sen = 3;


void readRawData(int *rawValue)
{
  Wire.beginTransmission(MPUAddr);
  Wire.write(0x3B);
```

```
    Wire.requestFrom(MPUAddr, nRegs * 2, true);
    Wire.endTransmission(true);

    int accel_x_h = Wire.read();int accel_x_l = Wire.read();
    int accel_y_h = Wire.read();int accel_y_l = Wire.read();
    int accel_z_h = Wire.read();int accel_z_l = Wire.read();
    int temp_h = Wire.read();int temp_l = Wire.read();
    int gyro_x_h = Wire.read();int gyro_x_l = Wire.read();
    int gyro_y_h = Wire.read();int gyro_y_l = Wire.read();
    int gyro_z_h = Wire.read();int gyro_z_l = Wire.read();

    rawValue[0] = accel_x_h <<8 | accel_x_l;
    rawValue[1] = accel_y_h <<8 | accel_y_l;
    rawValue[2] = accel_z_h <<8 | accel_z_l;
    rawValue[3] = temp_l <<8 |temp_h;
    rawValue[4] = gyro_x_h <<8 | gyro_x_l;
    rawValue[5] = gyro_y_h <<8 | gyro_y_l;
    rawValue[6] = gyro_z_h <<8 | gyro_z_l;

    }


float GetAcc_x(float *realValue)
{
    float fNormXZ = sqrt(pow(realValue[0],2) + pow(realValue[2],2));
    float result = atan(realValue[1] / fNormXZ) * R2D;
    return result;
    }

float GetAcc_y(float *realValue)
{
    float fNormYZ = sqrt(pow(realValue[1],2) + pow(realValue[2],2));
    float result =    atan( - realValue[0] / fNormYZ ) * R2D;
    return result;
    }


  void Calibration()
    {
        float Sum[7] = {0.0, 0.0, 0.0, 0.0, 0.0, 0.0, 0.0};

        for(int i = 0; i < nTimes ; i++)
        {
            int MPUvalues[nRegs];
```

```
         readRawData(MPUvalues);

         for(int j =0; j < nRegs; j++)
         {
            Sum[j] = Sum[j] + MPUvalues[j];
            }
         }

      for (int i = 0; i < nRegs ; i++)
      {
         CalibrationValue[i] = float (Sum[i] / nTimes);
         }



  }


void scaleFactor(int *before, float *after)
{
   for (int i = 0; i < 3; i++)
   {
      after[i] = (float)(before[i]);
   }


   for (int i = 4; i < 7; i++) {
      after[i] = (float)(before[i]) / 131.0f;
   }
}



void setup() {

   Serial.begin(19200);
   Wire.begin();
   Wire.beginTransmission(MPUAddr);
   Wire.write(0x6B);
   Wire.write(0);
   Wire.endTransmission(true);
   Calibration();
   double lastTime = millis();
```

```
    MySerial.begin(9600);




}

void loop()
{


    int readouts[nRegs];
    readRawData(readouts);
    double currentTime = millis();

    float x_accel = readouts[0];
    float y_accel = readouts[1];
    float z_accel = readouts[2];

/*
    Serial.print(F("accel x,y,z: "));
    Serial.print(x_accel, DEC);
    Serial.print(F(", "));
    Serial.print(y_accel, DEC);
    Serial.print(F(", "));
    Serial.print(z_accel, DEC);
    Serial.println(F(""));
  */


    float realVals[7];
    float baseVals[7];
    scaleFactor(readouts,realVals);
    scaleFactor(CalibrationValue,baseVals);

    float baseRoll = GetAcc_x(baseVals);
    float basePitch = GetAcc_y(baseVals);


    float Acc_x = GetAcc_x(realVals);
    float Acc_y = GetAcc_y(realVals);
    float Acc_z = 0;
    float dt = (currentTime - lastTime)/1000.0; //in second


    Gyro_x = realVals[4]*dt + Last_Angle_x ;
    Gyro_y = realVals[5]*dt + Last_Angle_y ;


/*
    Serial.println("*************************************");
    Serial.println("Gyro_x");
    Serial.println(Gyro_x);
    Serial.println("Acc_x");
    Serial.println(Acc_x);
    Serial.println("Last_Angle_x");
    Serial.println(Last_Angle_x);

    Serial.println("Gyro_y");
    Serial.println(Gyro_y);
    Serial.println("Acc_y");
    Serial.println(Acc_y);
    Serial.println("Last_Angle_y");
    Serial.println(Last_Angle_y);

*/




    int alpha = 0.96;
    float New_Angle_x = alpha * Gyro_x + (1 - alpha) * Acc_x ;
    float New_Angle_y = alpha * Gyro_y + (1 - alpha) * Acc_y;


    Last_Angle_x = New_Angle_x;
    Last_Angle_y = New_Angle_y;


    float outputRoll = -(New_Angle_x - baseRoll);
    float outputPitch = -(New_Angle_y - basePitch);

    if(outputRoll - last_outputRoll > sen || outputRoll - last_outputRoll < -sen || outputPitch -
last_outputPitch > sen || outputPitch - last_outputPitch < -sen)
    {
```

```
   delay(7.5);
   MySerial.println("R");
   MySerial.println(outputRoll);

   delay(7.5);


   MySerial.println("P");
   MySerial.println(outputPitch);
   delay(7.5);
   }

  else
  {
   last_outputRoll = outputRoll;
   last_outputPitch = outputPitch;
   }
}
```

This is the code for processing, some parts are modified based on opensource code. Code within comment are used for debugging.

```
import processing.serial.*;
import java.awt.Robot;
import java.awt.AWTException;
import java.awt.event.InputEvent;
import javax.swing.*;
import java.awt.MouseInfo;

Serial myPort;
String val;

char first;

float roll=0, pitch=0;
String Roll, Pitch;

float mousex,mousey;
int centerx = 540 ;
int centery = 960 ;

int xx,yy;
int nxx ;
int nyy ;

int sen = 9;


Robot cursor;

public void setup()
{
  try
  {
      cursor = new Robot(); //(EN) mouse = new Robot();
    }

  catch (AWTException e)
  {
      e.printStackTrace();
      println("error");
  }
```

```
   String portName = Serial.list()[2];

   myPort = new Serial(this, portName, 19200);

   myPort.clear();

   cursor.mouseMove(centerx,centery);

}



public void draw()
{

   if ( myPort.available() > 0)
   {
   val = myPort.readStringUntil('\n');                  // read it and store it in val
   }

   nxx = MouseInfo.getPointerInfo().getLocation().x;
   nyy = MouseInfo.getPointerInfo().getLocation().y;

//println(val);

//println(val.length());


if(val != null)
{
     first = val.charAt(0);

         if (first == 'R')
   {
      Roll = myPort.readStringUntil('\n');

      if(Roll != null)
      {roll = float(trim(Roll));}

      /*println("roll:"); //print it out in the console
      println(roll);*/
```

```
}

   else if (first == 'P')
{
   Pitch = myPort.readStringUntil('\n');
   if(Pitch != null)
   {pitch = float(trim(Pitch));}
   /*println("pitch:");
   println(pitch);*/
}


/*    else if (first == 'C')
{
   Pitch = myPort.readStringUntil('\n');
   println("count:");
   println(Pitch);
}*/


else
  {println("error");}


if(roll <= 50 || roll >= -50 || pitch <= 50 || pitch >= -50)
{

   mousex = map(pitch,-50,50,0,1919);
   mousey = map(roll,-50,50,0,1079);

   xx = int(mousex);
   yy = int(mousey);

   println("nxx");
   println(xx);
//println("xx");
//println(xx);


   println("nyy");
   println(yy);
//println("yy");
//println(yy);
```

```
      if( xx - nxx > sen || xx - nxx < -sen || yy - nyy > sen || yy - nyy < -sen)
        {
           cursor.mouseMove(xx,yy);
          }

      else
        {
         nxx = xx;
         nyy    = yy;

        }



   //cursor.mouseMove(xx,yy);

 }


 }

 }
```