UM-SJTU Joint Institute Undergraduate Programme
VG100 Project 2

Photographic Airship Based on Gesture Control

Team5: Crafters

Hongyi	Liu	516370910061
Jiangchen	Zhu	516370910183
Jiazheng	Xiong	516021910564
Linkai	Xue	516370910097
Yufan	Yue	516370910181

Instructors:
Prof. Yanfeng Shen
Prof. Cynthia Vagnetti

Date: 6 Aug 2017

Executive Summary

The remote-controlled cars and aircrafts have been introduced to people for decades and played an important role in not only daily-recreation but also professional application. However, the user experience for remote controllers still need improvement. It is tiring to hold it for more than an hour and too hard for amateurs to get familiar with. These two are born-with deficiency of traditional button&stick controlling method which left us little room for improvement. So we are looking for a new method of control to replace the traditional one: a commanding system based on gesture orders instead of button orders, which could simplify the process of controlling and solve the problems.

We apply the method on airship. The procedure of constructing the system consists of two parts: A gesture-telling glove and the multi-functional airship. The gesture-telling part is realized by two micro-accelerometers attached to the glove which detects the accelerations on x, y and z direction of two points. After processing the data with specific Arduino algorithm we can get a real-time detection on hand gestures. The airship is driven by four propellers(two horizontal and two vertical) and they are capable of all-direction movements. An information center consists of an Arduino board, a bread board, a transmission module and a liquid crystal display is installed on users' elbow to build the connection between the glove and the airship. We choose a high-power signal output module to ensure the steadiness of long-distance transmission. Also, a camera module was installed on the airship to record videos. With our product, even those who were not able to master the operation on remote controllers can enjoy the fun of flying unmanned aircrafts. Professional users do not need to keep an eye on their camera platform and can spare more time on targets with their hands freed.

Acknowledgements

Thank Prof.Yanfeng Shen, the instructor of VG100 for the technical part, for his interesting lectures, great patience and heartwarming encouragements.

Thank Prof.Cynthia Vagnetti, the instructor of VG100 for the technical communication part, for her vivid lectures, method of accuracy and priceless instructions.

Thank Yuhan Chen, Xucheng Ma, Weitao Sun, Yichen Yang and Cunzhi Gao, teaching assistants of VG100, for their priceless advice and kind instruction whenever we need help.

Thank all teammates in our group for their continuous hardwork and spirit of devoting during the project. This project would never be finished with the absence of anyone of us.

Thank all of our family members who gave us the warmest support when we encountered failures and challenges. We may not realize our design without their help and suggestion.

Thank Dr.Hongyan Wang, Associate Director of Product Development - Fluke Corporation at Danaher (China) Industrial Instrumentation Technologies R&D Co.Ltd. for her kind and enlightening instruction on the content and structure of our final report.

VG100 2016-17SU Section 2 Group5

Contents

List of Figures

1 Introduction

We are team Crafters(Fig.1), from University of Michigan-Shanghai Jiao Tong University Joint Institute(UM-SJTU JI). Our group members are Yue Yufan, Liu Hongyi, Xue Linkai, Xiong Jiazheng, Zhu Jiangchen. The address of our college is 800 Dongchuan Road, Minhang District, Shanghai, China.

Figure 1: Group members of team CRAFTERS

From left to right:Yufan Yue(Captain), Linkai Xue, Jiazheng Xiong, Hongyi Liu, Jiangchen Zhu

Our team consists of five freshmen with outstanding skills on programming and engineering among peers. In Project 1, we got full marks in the evaluation on weight of paper bridge and operating time of cart, which showed the power of teamwork between five students with a willing to make a difference. We aim to design and make a photographic airship based on gesture control in order to simplify the traditional button-stick operating system.

2 Problem

2.1 Statement of Problem

With the speedy development of unmanned aircraft and HD-camera technologies, people have cultivated great interest in aerial photography. But most of them are totally green-hands and can not master aircraft's control in a short time. We have done a survey to investigate the average time people used in getting familiar with traditional button-stick system.

According to Figure 2, only 26% of interviewees met little difficulty in the process of mastering the remote controller. And the rest 74% spent more than an hour on it. Thus there is a great need for a more simple operating system.

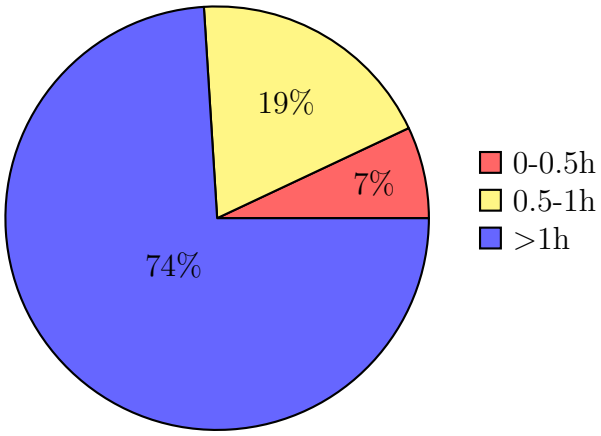

Figure 2: Statistics from the poll: How long did you spend to master a remote controller

2.2 Target Population

Since the gesture-telling system we designed could tell a large number of gestures serving as orders, users can easily command the airship with only one hand. The one-hand operating system enabled them spare more time on detailed tasks. So we listed three groups of target audiences.

1. Government force with need for long-term surveillance
2. Sport canals with need for matches recording
3. Amateur photographers with need for steady camera platform

2.3 Summary of Problem

All the remote controlling systems in the market have similar deficiency, we conclude them as three main problems as listed below and focus our design on solving them.

1. Tiring for long-time using
2. Complicated buttons
3. Complex to learn

3 Objectives

3.1 Methods to Address the Needs

As we all notice that the unmanned aerial planes are popular nowadays, they are used in many areas. However, unmanned aerial planes have disadvantages. Complicated commands and short cruising time are tow main shortcomings. Traditional unmanned aerial planes have the cruising time about 27 minutes. And the main controllers are hand shank or personal computer. Our goal is to lengthen the cruising time and simplify the commands to fit our customs's needs.

If we can reach our goals of our project, needs for long time video recording can be fitted. Customs such as football fans can use it to record a football match from sky.

When the commands are simplified, our project will be much friendly to children and old people. Both of the benefits will generalize the use of unmanned aerial recording devices, which could make people's life more convenient.

We applied the two methods in the Fig. 3 to reach these goals.

Figure 3: Airship is used to lengthen the cruising time and gesture control is used to simplify commands

3.2 Design

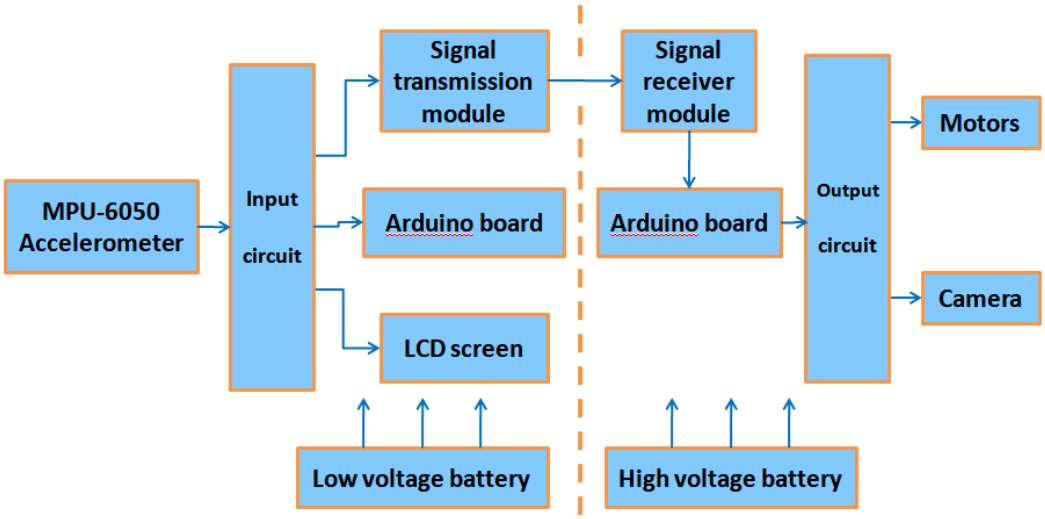

Figure 4: Block diagram of the system

The block diagram of the system is shown in Fig. 4.

In the gesture-telling glove and information center, accelerometers and LCD displayer are the direct devices that relates to users.As in the block diagram, accelerometers detects gestures and LCD displayer can show the latest commands to the users.

On the airship, motors, camera and the control center are the functional components of the system. The control center controls the motors and communicates with the information center of the glove.

4 Solution

A photographic airship based on gesture control is proposed, along with a handy operating design and a multifunctional airship that provides

1. A gesture-telling glove that is wearable and portable
2. A gesture design set that is greenhand-friendly
3. A quick-acting airship that can fly in any direction
4. A photographic system that can record videos in the air

4.1 Overview

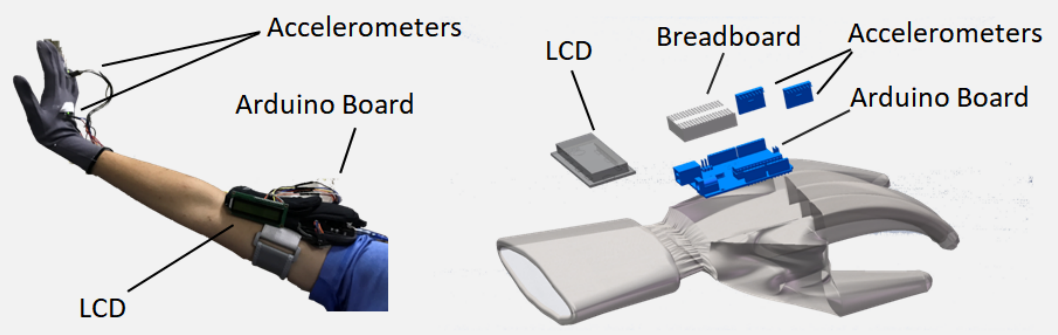

Figure 5: Concept diagram of gesture-telling glove and information center

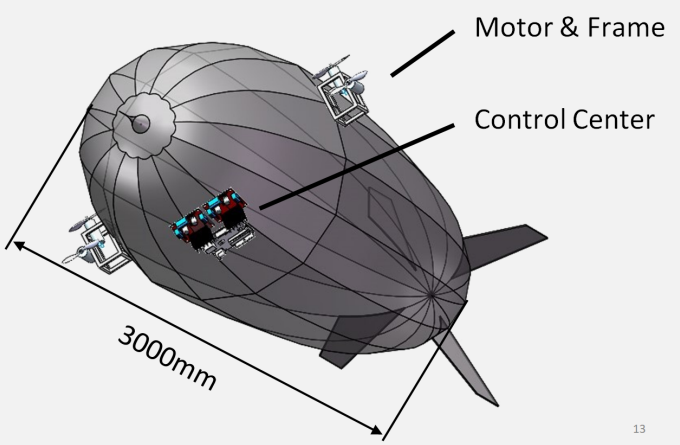

Figure 6: Concept diagram of airship

The overview of our project is presented by the two concept diagrams Fig. 5 and Fig. 6. The project consists of three major components: the gesture-telling glove, the information center and the photographic airship. The gesture telling glove is on the user's left hand and the information center will be placed on the user's left arm in an arm package.

4.2 Major Components and Individual Working Mechanism

4.2.1 Gesture-telling Glove

Accelerometer

The only and most important component in the gesture-telling glove system is the accelerometer. We have chosen MPU-6050 accelerometer module as shown in Fig.

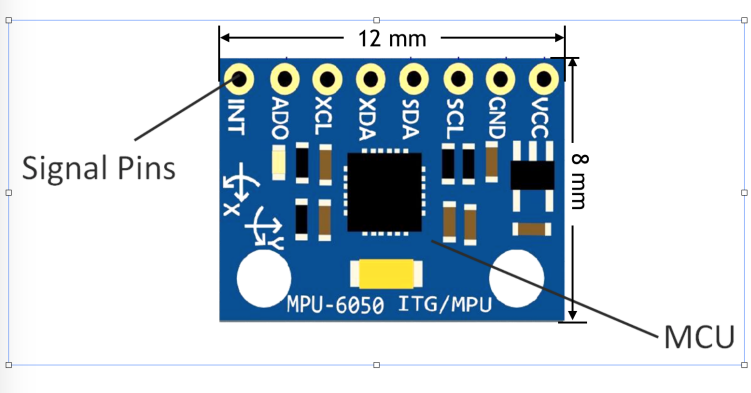

Figure 7: Photo of MPU-6050 accelerometer

7. The dimension is 4 $mm \times 4\ mm \times 0.9\ mm$, even smaller than a 1-yuan coin and the weight is less than 3 g. Therefore, it won't affect user's hand gestures and will provide a good user experience. The resolution of MPU-6050 module can reach 0.01 ^{o} and is adjustable. This enables a precise measurement of user's gestures. Besides, it adopts I2C protocol that can communicate steadily and effectively with Arduino board.

$$\vec{g} = x\hat{n_x} + y\hat{n_y} + z\hat{n_z} \tag{1}$$

As shown in Eq. 1, the accelerometer can detect the magnitude and orientation of gravitational and translate it into x, y, z coordinates with respect to the orientation of the accelerometer. Therefore we can tell the orientation of the accelerometer by calculating the relation between x, y, z values and eventually obtain the gesture of user's hand.

4.2.2 Information Center

Microcontroller

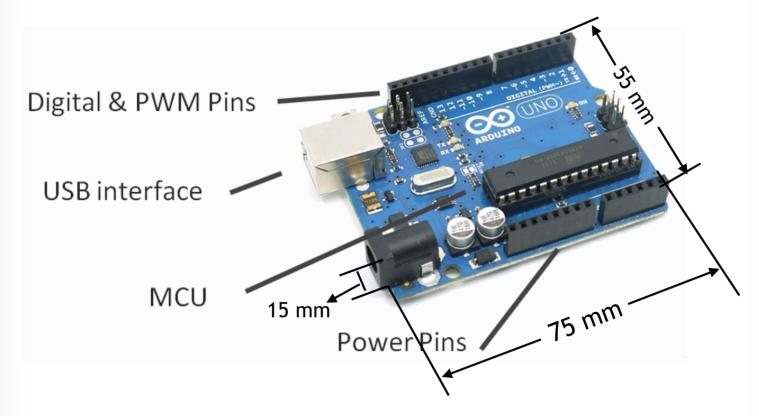

Figure 8: Photo of Arduino Uno R3

The "commander" of the information center is the Arduino Uno R3 board as shown in Fig. 8. It has a dimension of 75 $mm \times 55\ mm \times 15\ mm$ and weight about 50 g. This moderate size and weight will not give user any burden when carrying it on

his arm. Accompanied with low heat generation, it can ensure the freedom and safety of users. Besides, it has enough pins to accept and send signals.

Arduino will receive signal obtained by the accelerometer and process it with pre-installed algorithm. Then it will send commands to the airship to realize different functions.

Signal Transmission Module

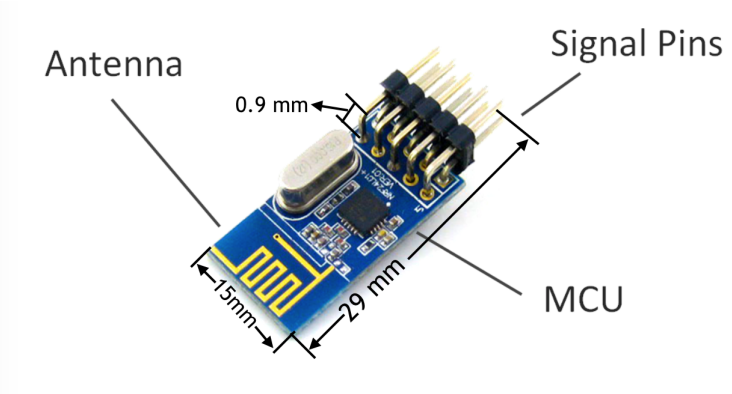

Figure 9: Photo of nrf24l01 2.4GHz FR transceiver/ receiver module

Fig. 9 shows the "messenger": nrf24l01 2.4GHz FR transceiver/ receiver module between the airship and the information center. "FR" stands for "frequency radio". This module has a dimension of 15 $mm \times 29\ mm \times 0.9\ mm$ (antenna included) and a weight of 2.5 g. The small size and light weight guarantee that it won't intervene the user's body movements. Its transmission distance can reach 44 meters without obstacles and 22 meters if there is something in the way as shown in Fig. 10. Besides, this module pair performs steadily. The long transmission distance and steady performance ensure a strong signal transmission from the information center to the airship.

Figure 10: Screenshot of transmission distance of nrf24l01 2.4GHz RF transceiver/ receiver module

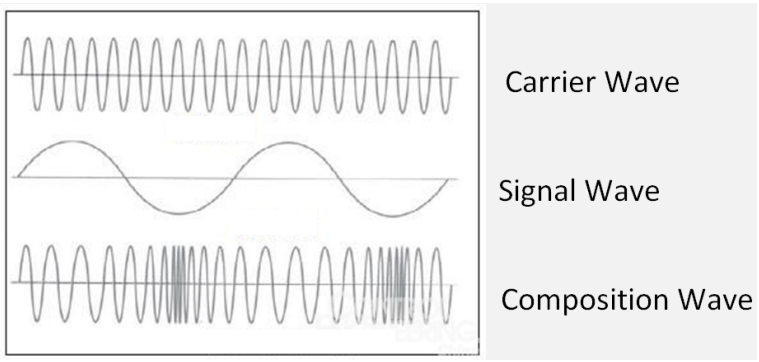

Figure 11: Working mechanism of FR module

As shown in Fig. 11, the frequency radio transceiver module will combine the signal wave with carrier wave to form a composite wave. Then the composite wave will be sent to the receiver module. The mixing method is based on the signal carrying property of different frequency in a composite wave. For example, when the signal is "LOW", the frequency of the composite wave will be high and vice versa. The frequency radio receiver module will decode the frequency wave, translate the signals and generate digital signals.

LCD Screen The LCD screen shown in Fig. 12 has a dimension of 80 $mm \times 36\ mm \times$

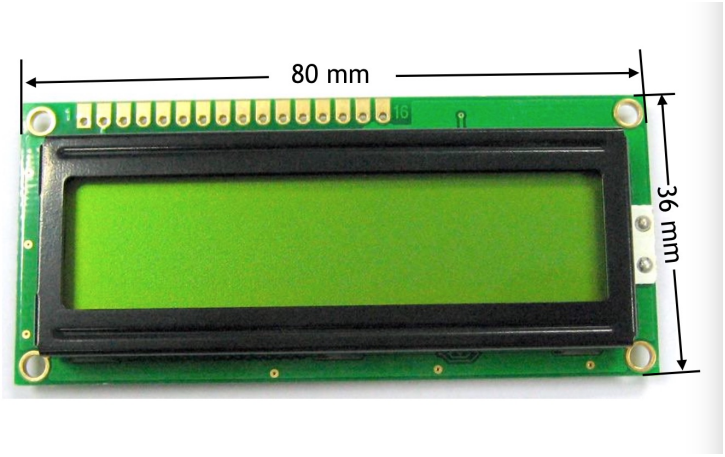

Figure 12: Photo of LCD1602 screen

13 mm and weight of about $50g$. This screen can show 16 characters * 2 rows at one time which is sufficient to give feedback to the user of the command they send to the airship. The screen is placed on the surface of the arm package in order to let the user see the feedback at the first sight.

4.2.3 Photographic Airship

Airship Body The airship is 3 meters long with maximum diameter of 1.6 meters. The general shape is paraboloid. The filling gas of the airship is helium which is the most common safe and inexpensive gas that has a density smaller than the air. The elevating force is provided by the density difference between air and helium.

$$F = (\rho_{air} - \rho helium)Vg = (1.293 - 0.1786) \times 9.8 \times 1.26 \approx 13.76\ N \qquad (2)$$

$$\Rightarrow V \approx \int \pi r^2 dz = \int_{-\frac{L}{2}}^{\frac{L}{2}} \pi (R - \frac{R}{(\frac{L}{2})^2} z^2)^2 dz = \frac{8\pi}{15} L R^2 \approx 1.26\ m^3 \tag{3}$$

$$\Rightarrow m = \frac{F - F_{ship}}{g} \approx 0.9\ kg \tag{4}$$

From Eq. 2 to 4, we have calculated that our airship can carry 0.9 kg of load which is sufficient for our electrical components.

Motor and Propeller

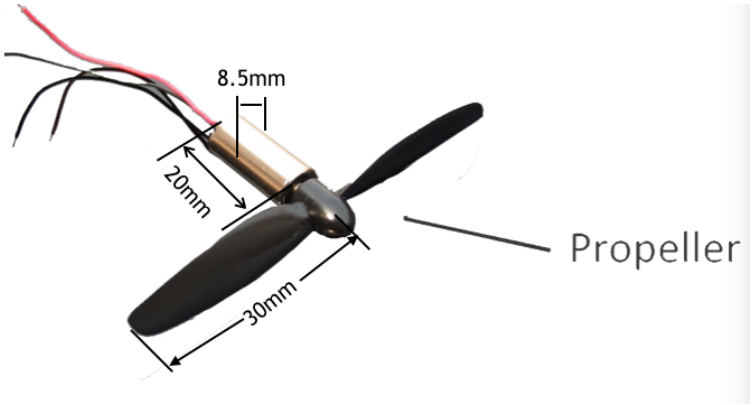

Figure 13: Photo of 820 drone motor with 6-cm diameter propeller

We have chosen 820 drone motor to drive our airship. The picture is shown in Fig. 13 820 drone motor is of cylindrical shape with diameter of 5 mm and height of 20 mm the motor shaft has diameter of 1 mm and length of 5mm. One motor weighs 3 g. The idling speed can reach 55000 rpm at 5 V and 45000 rpm at 3.7 V. Since the load capacity of airship is limited, it becomes especially important to minimize the weight and size of different components. This strong but light and small motor can meet our requirements perfectly. The propeller has diameter of 6 cm, which is small compared to other propellers.

$$kV = \frac{rpm}{V_{rated}} = \frac{45000\ rpm}{3.6\ V} = 12500\ rpm/V \tag{5}$$

In Eq. 5, kV value measures the balance between the motor's torque and its rotating speed. By referring to kV-propeller size table, we conclude that the higher kV value is, the smaller size of propeller we should choose to get maximum thrust. Besides, we noticed that this kV value is a lot higher than the data listed. Therefore, from 6-cm, 8-cm, 10-cm and 12-cm diameter propellers, we choose 6-cm propeller.

Driver Board We have chosen the most commonly used L298N driver board because of its small size, light weight and steady performance.

Microcontroller We have another Arduino UNO R3 board on the airship to analyze and control the airship. Since the board is introduced before, we will not repeat it.

Signal Receiving model There is an nrf24l01 2.4G RF receiver module (which is identical to the transceiver module) on the airship to receive signals from the information center.

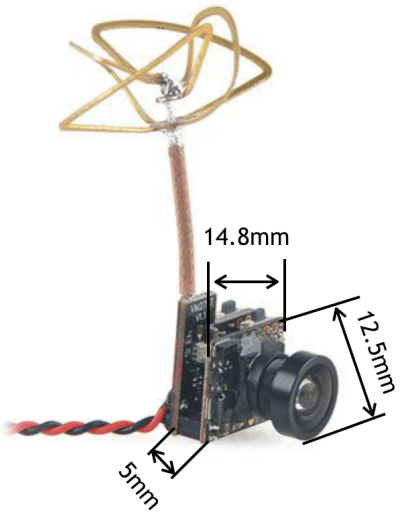

Figure 14: Photo of Crazepony 5.8G camera module

Camera Module

As shown in Fig. 14,this camera module have extremely small size of only about 14.8 $mm \times 12.5\ mm \times 5\ mm$ and light weight of only 3.6 g. Which is perfect for an airship. The image transmission distance can reach 100 m. Therefore, this module can ensure a good quality of photographic function of the airship.

4.3 Systematic Working Mechanism

The block diagram of the entire system is already shown in Fig. 4. When the user make a gesture, the accelerometer will get the data of the gesture and send the data to Arduino board. Then the data will be processed with pre-installed algorithms and be translated into new commands. The LCD screen will display the commands to give feedback to the user. The signal transmission module will realize signal transmission from the information center to the airship. On the airship, the signal receiver module will receive the signal, decode it and sent it to Arduino board. The Arduino board will analyze the command and give corresponding orders to the camera and motors. Finally, through the operation of camera and motors, the function of the airship can be realized.

5 Tasks

This part will cover the work done during building the prototype and constructing the whole system. As is shown in the flow chart below, the tasks by time sequence is divided into three parts: Design & Preparation, Test & Assembly, Improvement.

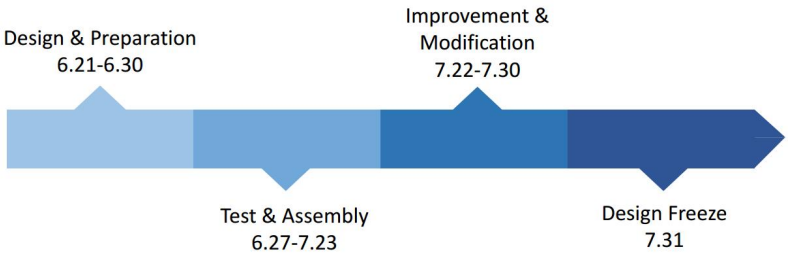

Figure 15: Flow chart of tasks for Project.2

5.1 Design & Preparation

To find the solution, we first began with design and other preparation work. We divided the system into three parts: hardware, software and mechanism and settled the target for each part. After division of the work, we all searched for related modules and purchased them for further testing.

5.1.1 Hardware

1.Fetch the signals of gesture To fetch the gesture we need to use sensors, related modules were accelerometers of different specification. Since the motion of airship involved only basic turning and movement in six directions, we decided to use MPU6050 accelerometer of 6-axis for gesture fetching.

2.Transmit command To get long transmission distance and a convenient matching process, we used a radio frequency module which transmit data easily and match automatically when it is powered. We chose the nrf24l01 module for its high holdings among engineers and simple library support.

3.Drive motors To get long cruising time the driving board must possess high circuit tolerance capacity. We decided to use Risym MOS driver board.

5.1.2 Software

1.Tell the gesture To process the signals we needed an algorithm that turned gestures into command for motors. We designed the basic gesture and took photo for future reference and discussed about the position of accelerometers on the hand.

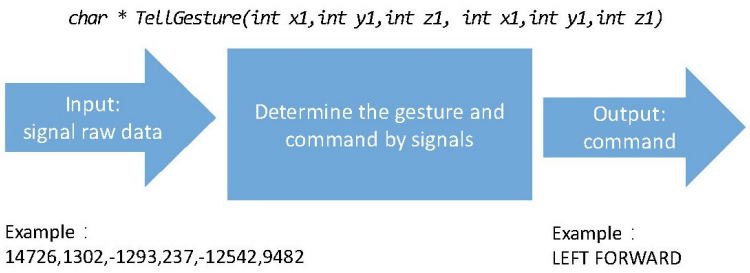

Figure 16: Function description of *TellGesture*

2.Output motor command To transfer the gesture into motor command, we programmed a related function to take input of gesture information and transfer it into pwm and digital output.

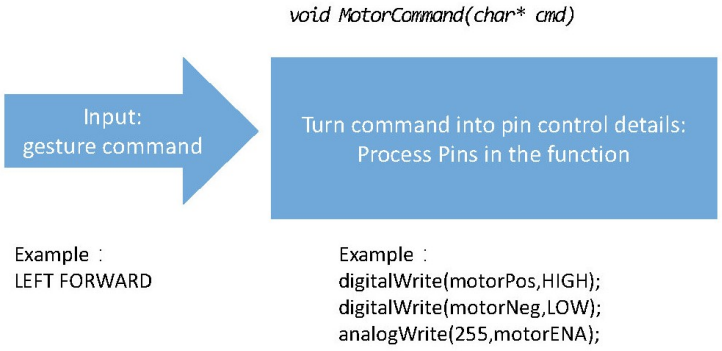

Figure 17: Function description of *MotorCommand*

5.1.3 Mechanism

1.Effectuate floating To make the airship suspend and balance the forces we used a 3-meter airship with attached balancing bag at the bottom. Hence we could adjust the moment and the resultant force.

Figure 18: Balancing bags with weights

2.Realize motion To provide sufficient thrust, we used a high RPM 820 motor with a motor frame. We used four incised Acrylic board to build a cuboid frame. This helped to adjust the position of motor and save enough space for air circulation.

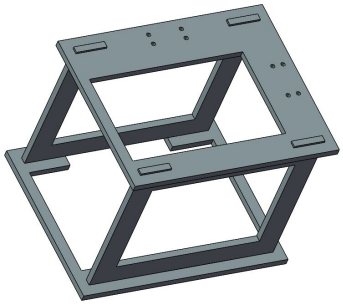

Figure 19: Design diagram for motor frame

5.2 Assembly & Test

To build the prototype we first tested the modules to understand its working mechanism. Next, we programmed the function, connected the circuit and assembled individual parts. Followed are some tasks during this procedure.

5.2.1 Accelerometer

For the accelerometer we tested its sensitivity and steadiness. This 6-axis accelerometer could reture six integers which denote the gravity acceleration component of x,y,z and angular acceleration towards x,y,z axis while rotating. The integer has high accuracy range from -16384 to +16384. After testing we decided to place the accelerometers on the back and the middle finger for the identification accuracy.

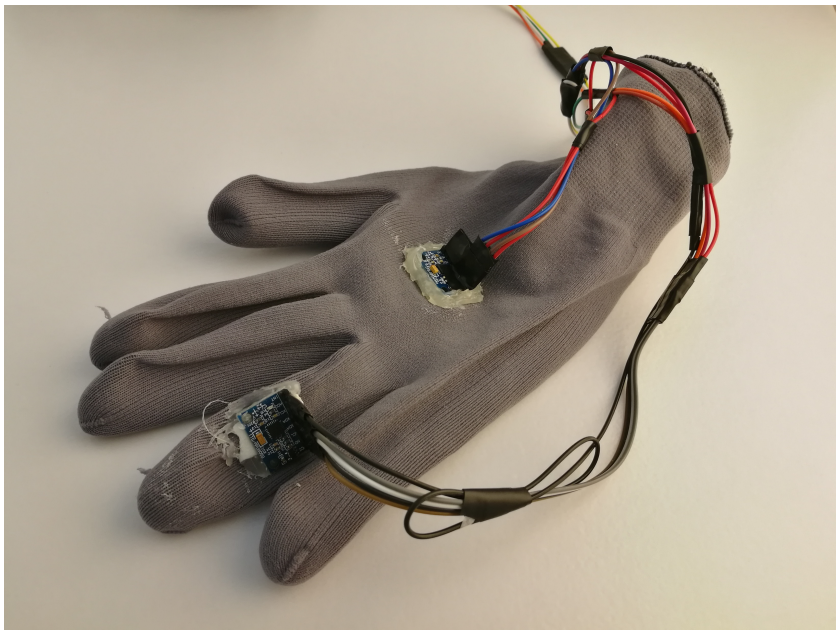

Figure 20: Installing accelerometers on the glove

5.2.2 Algorithm & Program

For the algorithm we first collected the raw signals of our designed gesture. We analyzed the feature of each gestures and devised corresponding threshold value. We tested its recognition precision for different people repetitively and modified the program. The table of gestures raw data for feature analysis is shown in the appendix.
For motor driving system, we connected the motor with the driving board and test their response to gesture command. We soon found the change of command and noise or fluctuation of signals may cause sudden turning of motor. For safety issue, we programmed another function to check whether the signal is identical to previous one and insert a time delay between each change to avoid drastic turning in the motor. The logic diagram of the program is shown below.

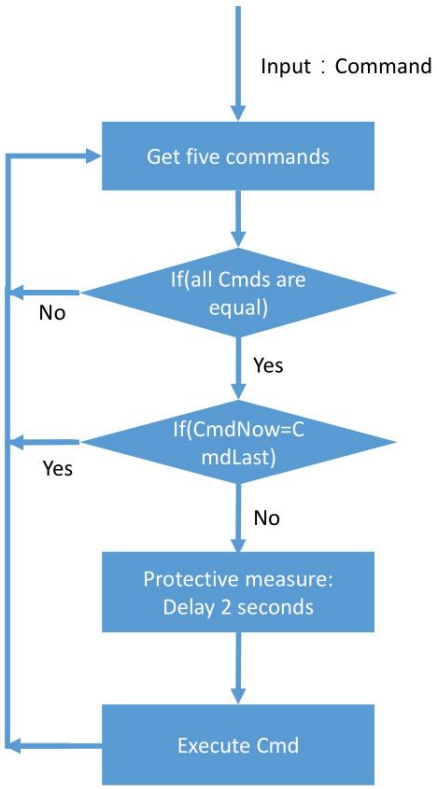

Figure 21: Logic flow of protective measures

5.2.3 Motor Frame & Balancing bags

For the frame we tested its stability under deformation and then attached it onto the airship. Since the frame only needed to bear little force, we made it thin and hollow to reduce its weight.
For the balance bags we needed to realize the balance in the vertical plane through the axis. Since the center of buoyancy was at the head of the airship, we used two bags at the front and back for balancing turn over and added weights on the motor frame for balancing rotation.

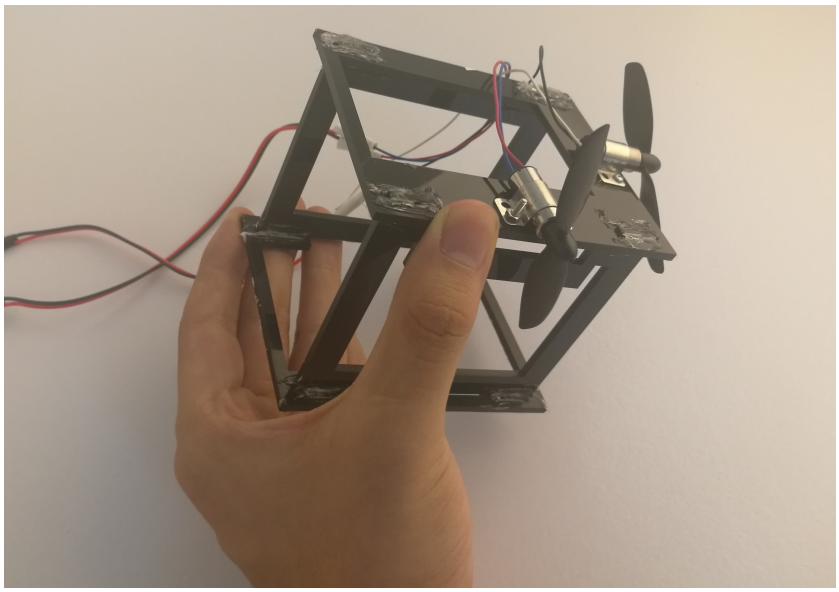

Figure 22: Frame's stability test under deformation

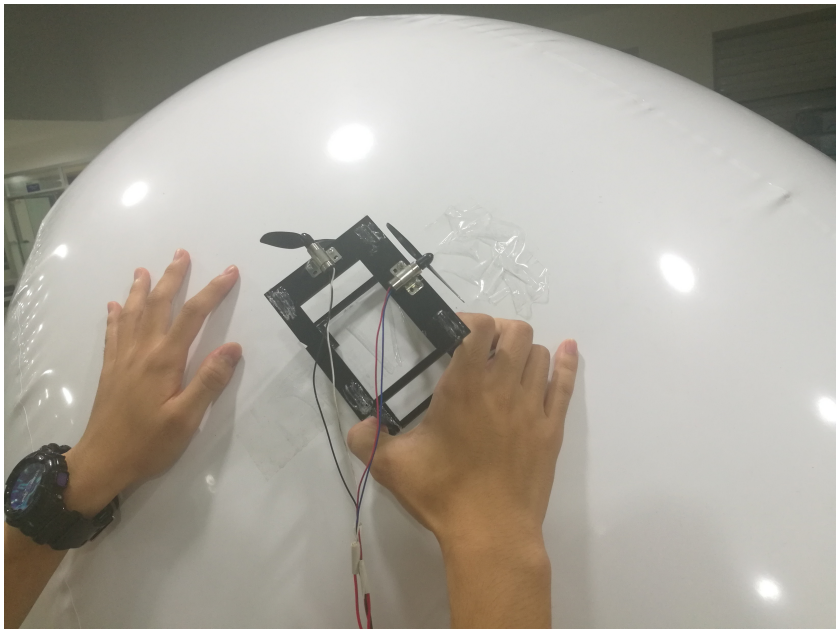

Figure 23: Installing motor frame

5.3 Improvement & Modification

To fulfill all the object and after debugging and modification of prototype we started the improvement part, we reconsidered the needs for our customer and made some improvement.

5.3.1 Feedback & Interaction System

To make the device more user friendly and also easier for maintenance, we added the feedback system. An LCD screen would print user's command and other information on the screen. This also helps engineers to test and repair the product.
To fit different personal judgment of what is vertical for users, we planned to add a reset

button for vertical rectification. The product would then adjust itself for users instead of users getting familiar with our product. This fits for our object of simple and intuitive control.The logic flow chart is shown below.

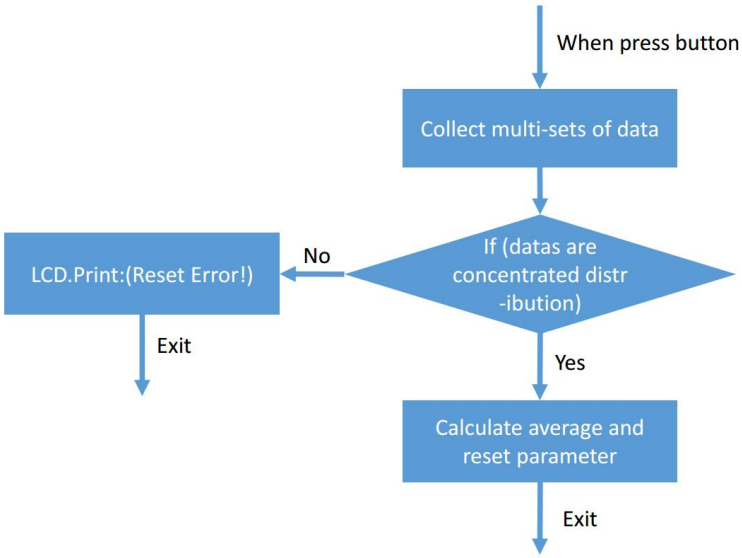

Figure 24: Logic flow of user's reset function

5.3.2 Transmission Ability

To suit for multiple needs of customers in different environment, we increased the transmission distance and enhance the ability of getting through obstacles and anti-interference. A more powerful transmission module is in our consideration.

Version	Transmission Distance in Open Area(m)	Cost(RMB)
nrf24l01	180-240	5.0
nrf24l01+signal amplification	1000-2000	11.2

Table 1: Comparison between transmission modules

5.3.3 Wearing-ability

To make the device wearable, we rearranged the modules of the glove. Previously, everything was attached on the glove and makes it hard to wear and limit the flexibility of the hand.Hence we separated everything except sensors from the glove and put them on the arm in a arm package.

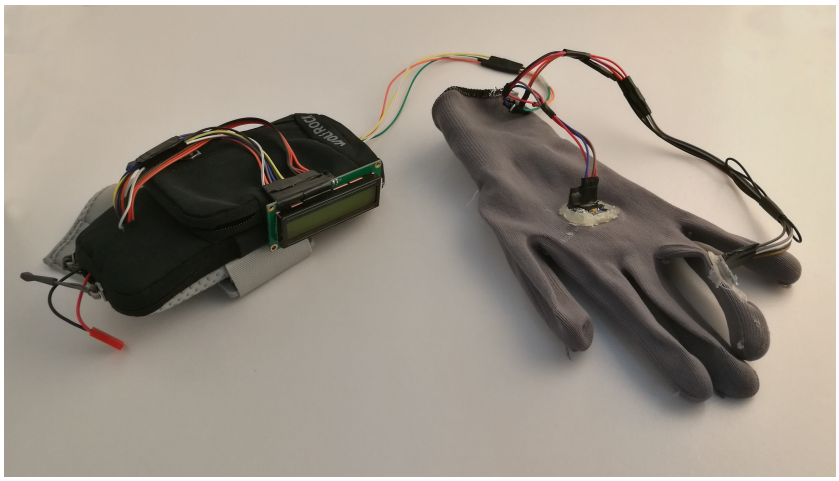

Figure 25: Glove after rearrangement

6 Schedule

This part shows the Gantt chart of our project. The chart is divided into three parts same as it is in the tasks: Design& Preparation, Test & Assembly , Improvement. Our progress remained steady during the whole process and we accomplished the project on time. The testing and making of glove took about two weeks, proceeded with about two weeks of building the airship. In the final week we improved the system to enhance its steadiness and stability.

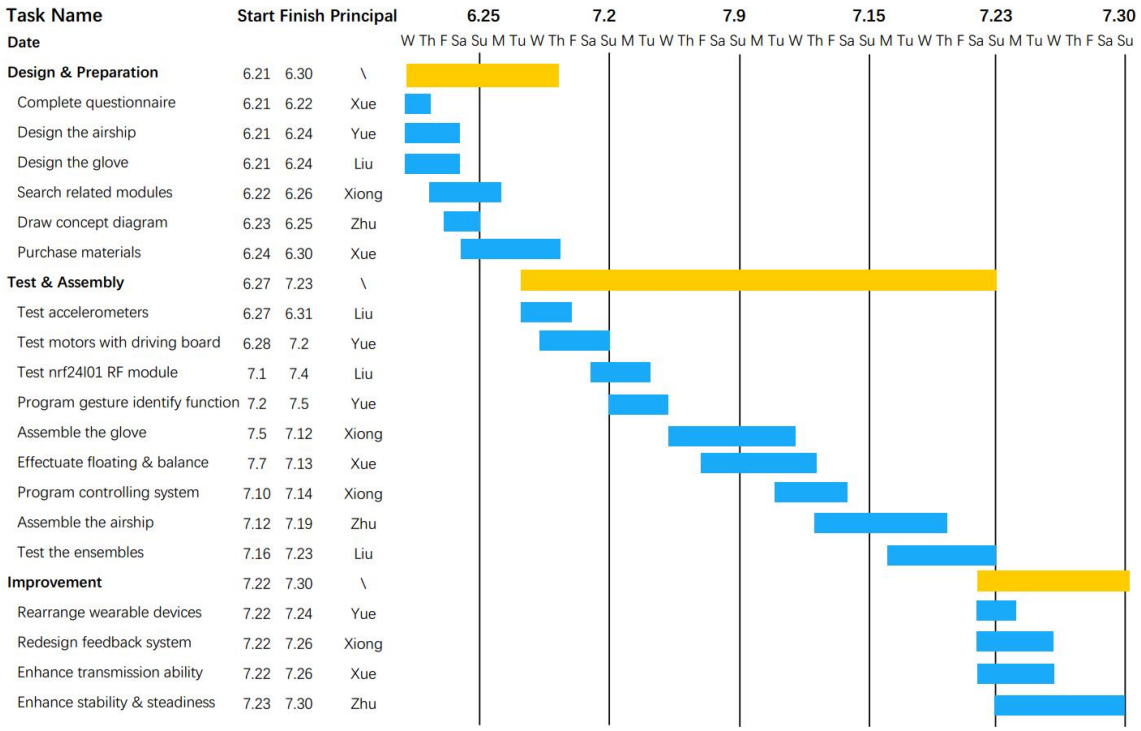

Figure 26: Gantt chart of Project.2

7 Budget

Our project can be divided into two parts, the glove and the airship. For the glove, we need two accelerometers, one Arduino board, one LCD, one breadboard, and one transmission module. To build the airship, we need one rechargeable airship, four motors, four driver boards, and one camera module.

The overall budget is 658.5 RMB, the detail of which is shown in Table 1. The airship takes up about 90% of the total cost, since the airship and the camera module are quite expensive. However, our project focuses mainly on the glove, which can detect one's gesture and can be applied to many other systems. In general, the glove only costs 68.5 RMB, which is inexpensive for ordinary people.

NO.	Item	Quantity	Unit Price (RMB)
Glove (68.5 RMB)			
1	MPU-6050 Accelerometer	2	15.0
2	Arduino Uno	1	25.5
3	LCD	1	7.0
4	Breadboard	1	1.0
5	Nrf-24101 Transmission Module	1	5.0
Airship (590.0 RMB)			
6	Rechargeable Airship	1	270.0
7	820 Drone Motor (with Propeller)	4	18
8	Risym MOS Driver Board	4	7.5
9	Camera Module	1	218.0

Table 2: The budget of our project.

8 Key Personnel

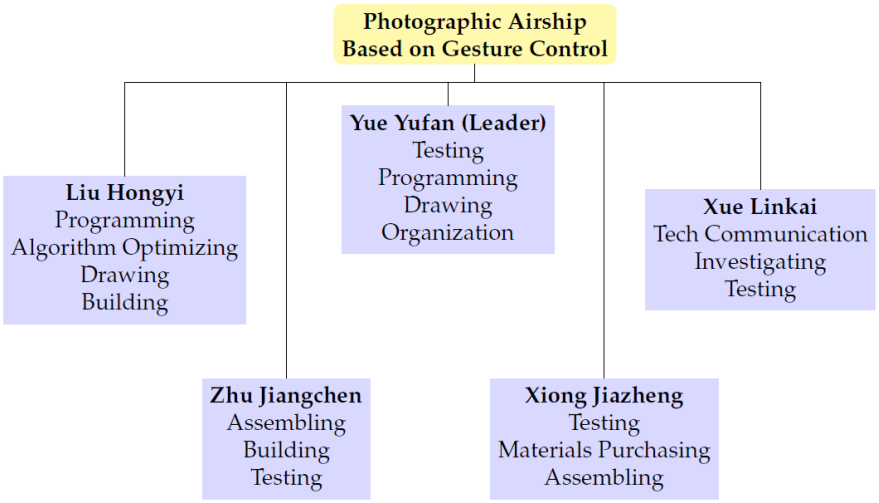

Figure 27: Tree diagram of key personnel

As shown in the tree above, Yue Yufan is our group leader. He organized the discussions and distributed the tasks . He was also in charge of testing the motors and coding programs for the glove and the airship. In addition, he drew concept diagrams using *AutoCAD*.

Liu Hongyi was in charge of programming of the transmission module and optimizing algorithm, which was the core part of our project. In addition, he also drew 3D concept diagrams using *Solidworks*.

Xue Linkai mainly worked on the technical communiation, he prepared for the presentation and wrote the reports. He also conducted a survey for our project. Moreover, he tested several components of the airship.

Zhu Jiangchen was in charge of assembling and building the airship. In addition, he helped Yue Yufan testing the motors and other components.

Xiong Jiazheng was in charge of buying materials on Taobao. Moreover, he also helped Yue Yufan testing the components and assembles the glove.

Note that although we had a clear distribution of tasks, we did not just accomplish our own task. We cooperated to reach higher efficiency. For many tasks we worked as a team, that is, each of us contributed to these tasks.

9 Shared Experience

9.1 Major Successes

1. The successful operation of accelerometers.

   We didn't spend a long period of time dealing with problems related with accelerometers. In fact, we succeeded the first time we tested it. This is mainly because our team has studied the accelerometers for a long time before installing and testing, so we had a rather comprehensive and complete understanding of its mechanism, method of installation, parameters and the related algorithm.

2. The good performance of signal transmission module.

   Again, the first set of transmission module appeared to be perfect for our design. Actually, our team has argued whether to use bluetooth module or this frequency radio module. Finally, after several day's discussion, we managed to take every important factors into account (the factors include the size, weight, transmission distance and stability) and carefully chose this module. As mentioned in previous sections, the transmission distance can reach 20 to 40 meters which is totally sufficient for our project.

We conclude that one should be well prepared before buying, installing or testing a component. The preparation should include reading articles about the mechanism, parameters and other possible knowledge about this component as well as discussing with peers to comprehend your ideas and concerns.

9.2 Major Obstacles

1. The choice of material of the airship.

   At first, we tried to use bamboo sticks as frames and plastic bags as airbags to make our airship. However, we soon found out that the bamboo frame was too deformable that it could easily damage the electrical components. Besides, we failed to manufacture air-tight plastic bags since we didn't have appropriate machines. Next, we turned to balloons as airbags. But we found that they couldn't provide enough a lifting force to support our airship. Finally we used a factor-produced PVC airship.

   The main reason is that we didn't do enough in-depth scientific calculations and theoretical analysis. If we calculated the buoyancy force, the size of the balloon and the weight of the system, we can immediately see that this solution is not going to work. Besides, we should not overestimate our manufacture skills.

2. The choice of motor.

   We first tried the 380-drone motor. Although it has a much stronger torque and stability than the current 820-drone motor, it is too heavy that the airship cannot carry 4 motors.

   We made this wrong choice at first because we neglected one critical criterion for an aircraft design —- light weight. After we consulted some specialists, we finally realized this problem and changed out plan.

In conclusion, theoretical analysis is a crucial part of designing. A team should first find a theoretical possible solution then deal with practical problems. In addition, a team is suggested to consult some specialists of this area to avoid the most common faults.

10 Future Improvements

Our products can still be improved a lot in the future.

1. We will install a servo system to enable rotation of the camera. In this way, the user can get a better view.
2. We will enhance our algorithm to realize a more smart motion of the airship. For example, after a turning command, we will make the motors rotate in the opposite direction to stop the airship's rotation due to inertia. Besides, we will add a reset button to easily reset the airship to rest.
3. We will install more accelerometers on the glove and better our algorithm to detect more subtle gestures and make more complex movements.
4. We will install a wind-preventing system to enhance the stability of the airship.
5. We will further improve the signal transmission module to enable a transmission distance of at least 5km in order to be used in real life.

11 References

[1] jrowberg, 2017. "i2cdevlib," https://github.com/jrowberg/i2cdevlib
[2] aaronds, 2012. "arduino-nrf24l01," https://github.com/aaronds/arduino-nrf24l01

Appendix A Codes

A.1 receive.ino

This is the code for the signal receiver module.

```
#include "SPI.h"
#include "Mirf.h"
#include "nRF24L01.h"
#include "MirfHardwareSpiDriver.h"
#include <Wire.h>
int value;
void setup()
{
        Wire.begin(2);
        Wire.onRequest(motorcmd);
        Serial.begin(38400);
        Mirf.spi = &MirfHardwareSpi;
        Mirf.init();
        Mirf.csnPin = 10;
        Mirf.cePin = 9;
        Mirf.setRADDR((byte *)"WYDSS"); //Set our own receiving address with five letters
        Mirf.payload = sizeof(value);
        Mirf.channel = 90;    //Decide which channel to use
        Mirf.config();
        Mirf.config();
        Serial.println("Listening...");    //Begin to monitor on recived data
}
void loop()
{
        if (Mirf.dataReady()) {  //Output the received data as soon as the programme is activated
                Mirf.getData((byte *)&value);
                Serial.print("Got data: ");
                Serial.println(value);
        }
        Wire.onRequest(motorcmd);
        delay(200);
}
void motorcmd()
{
        Wire.write(value);
        // Wire.write("Hello "); // Output six char to the Information center
}
```

A.2 Relay.cpp

```
#include"Relay.h"
#include<Arduino.h>
#include <stdio.h>
#include <string.h>
#include <inttypes.h>
using namespace std;
Relay::Relay()
{
        this->in1=0;
        this->in2=0;
}
Relay::Relay(int in1,int in2)
{
        this->in1=in1;
        this->in2=in2;
}
int Relay::pos()
{
        digitalWrite(this->in1, HIGH);
        digitalWrite(this->in2,LOW);
}
int Relay::neg()
{
        digitalWrite(this->in1, LOW);
        digitalWrite(this->in2,HIGH);
}
int Relay::off()
{
        digitalWrite(this->in1, LOW);
        digitalWrite(this->in2, LOW);
}
```

A.3 Relay.h

```
#pragma once
#ifndef TEST_H
#define TEST_H
class Relay
{
public:
        Relay();
        Relay(int in1, int in2);
        int in1;// signal pin number
        int in2;// signal pin number
```

```
        int pos();// switch to positive direction mode
        int neg();// switch to negative direction mode
        int off();// switch to no turning mode
};
#endif
```

A.4 Relay_arduino.ino

This is the code controlling the motor.

```
#include"Relay.h"
#include <Wire.h>
int loopnum = 0;// the current number of loop
int movement = 0;// the received movement command
int isStby();// an enhancement of the safty system.
//i.e. increase the sensitivity of detecting signals while
    still preventing the motors from suddenly turning in the
    opposite direction
int tt;

Relay r1(2, 3);
Relay r2(6, 7);
Relay r3(9, 8);
Relay r4(5, 4);// define the signal pins for the four motors

#define number 4 // number of motors
int value1, value2, value3, value4, value5;// receive five
    commands one time to eliminate random error
int cmd[number];
int reccmd[4];// an array to store the raw received command

void setup()
{
  // put your setup code here, to run once:
  pinMode(r1.in1, OUTPUT);
  pinMode(r1.in2, OUTPUT);
  pinMode(r2.in1, OUTPUT);
  pinMode(r2.in2, OUTPUT);
  pinMode(r3.in1, OUTPUT);
  pinMode(r3.in2, OUTPUT);
  pinMode(r4.in1, OUTPUT);
  pinMode(r4.in2, OUTPUT);
  digitalWrite(r1.in1, LOW);
  digitalWrite(r1.in2, LOW);
  digitalWrite(r2.in1, LOW);
  digitalWrite(r2.in2, LOW);
  digitalWrite(r3.in1, LOW);
  digitalWrite(r3.in2, LOW);
```

```
36    digitalWrite(r4.in1, LOW);
37    digitalWrite(r4.in2, LOW);
38    Wire.begin();
39    Serial.begin(38400);
40    Serial.println("Begin!");
41  }
42
43  void loop() {
44    Serial.println(++loopnum);
45    Wire.requestFrom(2, 1);// get command from slave arduino
          board
46
47    while (Wire.available())// read the first raw command
48    {
49      value1 = Wire.read();
50      Serial.print(value1);
51      Serial.print("\t");
52    }
53    delay(200);// get a new raw command every 0.2s
54    Wire.requestFrom(2, 1);
55
56    while (Wire.available())
57    {
58      value2 = Wire.read();
59      Serial.print(value2);
60      Serial.print("\t");
61    }
62    delay(200);
63
64    Wire.requestFrom(2, 1);
65
66    while (Wire.available())
67    {
68      value3 = Wire.read();
69      Serial.print(value3);
70      Serial.print("\t");
71    }
72    delay(200);
73
74    Wire.requestFrom(2, 1);
75
76    while (Wire.available())
77    {
78      value4 = Wire.read();
79      Serial.print(value4);
80      Serial.print("\t");
81    }
82    delay(200);
```

```
83
84    Wire.requestFrom(2, 1);
85
86    while (Wire.available())
87    {
88      value5 = Wire.read();
89      Serial.print(value5);
90      Serial.print("\t");
91    }
92    delay(200);
93
94    if (value1 == value2 && value1 == value3 && value1 ==
          value4 && value1 == value5)
95      // if these 5 commands are the same, execute the command
            , otherwise do nothing
96    {
97      if (movement != value1 && !isStby() ) {
98        reccmd[0] = 0;
99        reccmd[1] = 0;
100       reccmd[2] = 0;
101       reccmd[3] = 0;
102       delay(2000);
103     }
104     movement = value1;
105     reccmd[0] = 0 + value1 / 100;
106     reccmd[1] = 0 + (value1 - (value1 / 100) * 100) / 10;
107     reccmd[2] = 0 + value1 - (value1 / 10) * 10;
108     reccmd[3] = 0;
109     Serial.println( );
110   }
111   else {
112     reccmd[0] = 0;
113     reccmd[1] = 0;
114     reccmd[2] = 0;
115     reccmd[3] = 0;
116   }
117   Serial.println();
118   Serial.print(reccmd[0]); Serial.print(reccmd[1]); Serial.
          print(reccmd[2]); Serial.print(reccmd[3]);
119   Serial.println();
120   // now we translate the raw commands into motor movements
121   if (reccmd[0] == 0 && reccmd[1] == 0 && reccmd[2] == 0 &&
          reccmd[3] == 0) {
122     cmd[0] = 0;
123     cmd[1] = 0;
124     cmd[2] = 0;
125     cmd[3] = 0;
126   }
```

```
127     else if (reccmd[2] != 0) {
128       cmd[0] = reccmd[2];
129       cmd[1] = reccmd[2];
130       cmd[2] = 0;
131       cmd[3] = 0;
132     }
133     else {
134       if (reccmd[0] == 0) {
135         cmd[0] = 0;
136         cmd[1] = 0;
137         cmd[2] = reccmd[1];
138         cmd[3] = reccmd[1];
139       }
140       else {
141         if (reccmd[0] == 1 && reccmd[1] == 0) {
142           cmd[0] = 0;
143           cmd[1] = 0;
144           cmd[2] = 1;
145           cmd[3] = 2;
146         }
147         else if (reccmd[0] == 1 && reccmd[1] == 1) {
148           cmd[0] = 0;
149           cmd[1] = 0;
150           cmd[2] = 1;
151           cmd[3] = 0;
152         }
153         else if (reccmd[0] == 1 && reccmd[1] == 2) {
154           cmd[0] = 0;
155           cmd[1] = 0;
156           cmd[2] = 0;
157           cmd[3] = 2;
158         }
159         else if (reccmd[0] == 2 && reccmd[1] == 0) {
160           cmd[0] = 0;
161           cmd[1] = 0;
162           cmd[2] = 2;
163           cmd[3] = 1;
164         }
165         else if (reccmd[0] == 2 && reccmd[1] == 1) {
166           cmd[0] = 0;
167           cmd[1] = 0;
168           cmd[2] = 0;
169           cmd[3] = 1;
170         }
171         else if (reccmd[0] == 2 && reccmd[1] == 2) {
172           cmd[0] = 0;
173           cmd[1] = 0;
174           cmd[2] = 2;
```

```
175             cmd[3] = 0;
176           }
177         }
178       }
179
180       // implement the commands
181       if (cmd[0] == 0) {
182         r1.off();
183         Serial.print("Relay #"); Serial.print("1"); Serial.print
              (" is off!"); Serial.println(" ");
184       }
185       else if (cmd[0] == 1) {
186         r1.pos();
187         Serial.print("Relay #"); Serial.print("1"); Serial.print
              (" is pos!"); Serial.println(" ");
188       }
189       else if (cmd[0] == 2) {
190         r1.neg();
191         Serial.print("Relay #"); Serial.print("1"); Serial.print
              (" is neg!"); Serial.println(" ");
192       }
193
194       if (cmd[1] == 0) {
195         r2.off();
196         Serial.print("Relay #"); Serial.print("2"); Serial.print
              (" is off!"); Serial.println(" ");
197       }
198       else if (cmd[1] == 1) {
199         r2.pos();
200         Serial.print("Relay #"); Serial.print("2"); Serial.print
              (" is pos!"); Serial.println(" ");
201       }
202       else if (cmd[1] == 2) {
203         r2.neg();
204         Serial.print("Relay #"); Serial.print("2"); Serial.print
              (" is neg!"); Serial.println(" ");
205       }
206       if (cmd[2] == 0) {
207         r3.off();
208         Serial.print("Relay #"); Serial.print("3"); Serial.print
              (" is off!"); Serial.println(" ");
209       }
210       else if (cmd[2] == 1) {
211         r3.pos();
212         Serial.print("Relay #"); Serial.print("3"); Serial.print
              (" is pos!"); Serial.println(" ");
213       }
214       else if (cmd[2] == 2) {
```

```
215       r3.neg();
216       Serial.print("Relay #"); Serial.print("3"); Serial.print
              (" is neg!"); Serial.println(" ");
217     }
218     if (cmd[3] == 0) {
219       r4.off();
220       Serial.print("Relay #"); Serial.print("4"); Serial.print
              (" is off!"); Serial.println(" ");
221     }
222     else if (cmd[3] == 1) {
223       r4.pos();
224       Serial.print("Relay #"); Serial.print("4"); Serial.print
              (" is pos!"); Serial.println(" ");
225     }
226     else if (cmd[3] == 2) {
227       r4.neg();
228       Serial.print("Relay #"); Serial.print("4"); Serial.print
              (" is neg!"); Serial.println(" ");
229     }
230     Serial.println("end a loop.");
231   }
232
233   int isStby() {
234     int now;
235     now = millis();
236     if (now - millis() >= 2000 && movement == 0) {
237       tt = now;
238       return 1;
239     }
240     else {
241       if (movement != 0);
242       tt = now;
243     }
244     return 0;
245   }
```

A.5 test_p2_transmission_final.ino

This is the code for the signal transmission module.

```
1  // Arduino Wire library is required if I2Cdev
       I2CDEV_ARDUINO_WIRE implementation
2  // is used in I2Cdev.h
3
4  // I2Cdev and MPU6050 must be installed as libraries, or
       else the .cpp/.h files
5  // for both classes must be in the include path of your
       project
```

```
6
7  // class default I2C address is 0x68
8  // specific I2C addresses may be passed as a parameter here
9  // AD0 low = 0x68 (default for InvenSense evaluation board)
10 // AD0 high = 0x69
11
12 //SCL-A5 SDA-A4 ADO(hand)-7 ADO(finger)-8 GND-GND VC-3.3v
13 //VSS,RW-GND VDD-5v V0-10 RS-7 E-6 D4,D5,D6,D7-5,4,3,2
14 //A(VCC)-3.3v B(CSN)-9 C(MOSI)-11 D-NULL E(GND)-GND F(CE)-8
      G(SCK)-13 H(MISO)-12
15
16 #include "I2Cdev.h"
17 #include "MPU6050.h"
18 #include "Wire.h"
19 #include <LiquidCrystal.h>
20 #include <SPI.h>
21 #include "Mirf.h"
22 #include "nRF24L01.h"
23 #include "MirfHardwareSpiDriver.h"
24
25 int value;
26
27 #define luminance 10
28
29 MPU6050 accelgyro(0x68);
30 MPU6050 accelgyro2(0x69);
31 LiquidCrystal lcd(7, 6, 5, 4, 3, 2);
32 int16_t ax_hand, ay_hand, az_hand, ax_finger, ay_finger,
      az_finger;
33 int16_t gx_hand, gy_hand, gz_hand, gx_finger, gy_finger,
      gz_finger;
34 char cmd[] = "ABCD";
35 bool blinkState = false;
36
37 void sensor_get();
38 void serial_print();
39 void lcd_print();
40 void function(int x1, int y1, int z1, int x2, int y2, int z2
      );
41 char isaction(int x1, int y1, int z1, int x2, int y2, int z2
      );
42 char UoD(int x1, int y1, int z1, int x2, int y2, int z2);
43 char FoB(int x1, int y1, int z1, int x2, int y2, int z2);
44 char LoR(int x1, int y1, int z1, int x2, int y2, int z2);
45 void send2boat();
46 void setup() {
47
48   delay(1000);
```

```
49    Wire.begin();
50    pinMode(luminance, OUTPUT);
51    lcd.begin(16, 2);
52    lcd.print("Your command:");
53    analogWrite(luminance, 140);
54
55    Mirf.cePin = 8;                    //Set the CE pin as D8
56    Mirf.csnPin = 9;           //Set the CE pin as D9
57    Mirf.spi = &MirfHardwareSpi;
58    Mirf.init();
59
60    Mirf.setRADDR((byte *)"SSDYW"); //Set the own sending
          address with five letters
61    Mirf.payload = sizeof(value);
62    Mirf.channel = 90;                 //Set the signal channel
          used
63    Mirf.config();
64
65    Serial.begin(38400);
66    Serial.println("Initializing I2C devices...");
67    accelgyro.initialize();
68    delay(1000);
69    accelgyro2.initialize();
70
71    Serial.println("Testing device connections...");
72    Serial.println(accelgyro.testConnection() ? "MPU6050
          connection successful" : "MPU6050 connection failed");
73
74  }
75  void loop() {
76
77    sensor_get(); //get value
78    function(ax_hand, ay_hand, az_hand, ax_finger, ay_finger,
          az_finger);
79    serial_print(); //print in the serial
80
81    lcd_print();     // print in the lcd
82
83    send2boat();
84
85
86    delay(200);//time control
87  }
88
89  void send2boat() {
90    Mirf.setTADDR((byte *)"WYDSS");//Set the receiving address
91    value=0;
92    if(cmd[0]=='L') value+=100;
```

```
93    else if(cmd[0]== 'R') value+=200;
94    if (cmd[1]== 'F') value+=10;
95    else if(cmd[1]== 'B') value+=20;
96    if (cmd[2]== 'U') value+=1;
97    else if(cmd[2]== 'D') value+=2;
98    Serial.println(value);
99    Mirf.send((byte *) &value);                    //
100   while(Mirf.isSending()) delay(1);             //Exit the loop
          until transmission confirmed
101 }
102
103 void sensor_get() {
104   accelgyro.getMotion6(&ax_hand, &ay_hand, &az_hand, &
          gx_hand, &gy_hand, &gz_hand);
105   accelgyro2.getMotion6(&ax_finger, &ay_finger, &az_finger,
          &gx_finger, &gy_finger, &gz_finger);
106 }
107
108 void serial_print() {
109   Serial.print("hand:");
110   Serial.print("a/g:\t");
111   Serial.print(ax_hand);
112   Serial.print("\t");
113   Serial.print(ay_hand);
114   Serial.print("\t");
115   Serial.print(az_hand);
116   Serial.println("\t");
117
118   Serial.print("Finger:");
119   Serial.print("a/g:\t");
120   Serial.print(ax_finger);
121   Serial.print("\t");
122   Serial.print(ay_finger);
123   Serial.print("\t");
124   Serial.println(az_finger);
125
126   Serial.print("Your command is:");
127   Serial.print(cmd[0]);
128   Serial.print(cmd[1]);
129   Serial.print(cmd[2]);
130   Serial.print(cmd[3]);
131   Serial.println(cmd[4]);
132 }
133
134 void lcd_print() {
135   lcd.setCursor(0, 1);
136   lcd.print(cmd[0]);
137   lcd.print(cmd[1]);
```

```
138     lcd.print(cmd[2]);
139     lcd.print(cmd[3]);
140     lcd.print(" t=");
141     lcd.print(millis() / 1000);
142     lcd.print("  +1s     ");
143 
144 }
145 
146 void function(int x1, int y1, int z1, int x2, int y2, int z2
        ) {// x1, y1, z1 is the data of the finger sensor; x2, y2,
         z2 is the data of the handback sensor
147   static char value[5];
148   if (isaction(x1, y1, z1, x2, y2, z2) == '0') { // isaction
        (...) detects whether there is action. '0' for no action
149     value[0] = '0'; // set all to '0'
150     value[1] = '0';
151     value[2] = '0';
152     value[3] = '0'; // the last (No.4) digit controls
          whether it will move. '0' for static.
153     value[4] = '\0'; // protect the string
154     for (int i = 0; i < 5; i++)
155       cmd[i] = value[i];
156     return ;// function returns.
157   }
158   else if (isaction(x1, y1, z1, x2, y2, z2) == '1') {
159     value[4] = '\0'; // protect the string
160     value[3] = '1'; // the last digit is '1', means there is
          action
161     if (UoD(x1, y1, z1, x2, y2, z2) == 'U') { // UoD(...)
          detects whether it will move up or down. 'U' for up, '
         D'for down, '0' for static
162       value[2] = 'U';
163       value[0] = '0'; // don't move left or right
164       value[1] = '0'; // don't move forward or backward
165       for (int i = 0; i < 5; i++)
166         cmd[i] = value[i];
167       return;// when moving up or down, the airship don't
            move forward/ backward/ left/ right.
168     }
169     else if (UoD(x1, y1, z1, x2, y2, z2) == 'D') {
170       value[2] = 'D';
171       value[0] = '0';
172       value[1] = '0';
173       for (int i = 0; i < 5; i++)
174         cmd[i] = value[i];
175       return;
176     }
177     else { // the ship doesn't move up or down
```

```
178         value[2] = '0'; // set the digit to '0'
179         if (FoB(x1, y1, z1, x2, y2, z2) == 'F') value[1] = 'F'
                ; // forward
180         else if (FoB(x1, y1, z1, x2, y2, z2) == 'B') value[1]
                = 'B'; // backward
181         else if (FoB(x1, y1, z1, x2, y2, z2) == '0') value[1]
                = '0'; // static
182         if (LoR(x1, y1, z1, x2, y2, z2) == 'L') value[0] = 'L'
                ; // left
183         else if (LoR(x1, y1, z1, x2, y2, z2) == 'R') value[0]
                = 'R'; // right
184         else if (LoR(x1, y1, z1, x2, y2, z2) == '0') value[0]
                = '0'; // static
185         for (int i = 0; i < 5; i++)
186           cmd[i] = value[i];
187         return;
188       }
189     }
190 }
191
192 /*
      ************************************************************
      */
193 // the following functions are position functions of hands.
      Please refer to photos.
194
195 char UoD(int x1, int y1, int z1, int x2, int y2, int z2) {
196   //|x1|,|x2| are large; y1,y2,z1,z2 are small;
197   //see photos
198   if (x1 >= 10000 && x2 >= 10000) return 'U'; // up
199   else if (x1 >= 10000 && x2 <= -10000) return 'D';// down
200   else return '0';// not move up or down
201 }
202
203 char FoB(int x1, int y1, int z1, int x2, int y2, int z2) {
204   if (x1 < -8000 || x1 > 8000) return '0'; // axis of hand
        won't be horizontal
205   else if (z2 < 2000 && z2 > -2000) return '0'; // palm won'
        t be vertical
206   if (y1 < -13000 || y1 > 5000) return '0'; // axis of hand
        won't be horizontal
207   else if (((z1 > 0 && z2 > 0) || (z1 < 0 && z2 < 0)) && z1
        - z2 <= 3000 && z2 - z1 <= 3000) return 'F'; // palm is
        open --> finger and handback is in a line the same
        direction.
208   else if (((z1 > 0 && z2 < 0) || (z1 < 0 && z2 > 0)) && z1
        + z2 <= 5000 && z1 + z2 >= -4000 || (z1 < -10000 && z1 +
         z2 > 0) || (z1 > 17000 && z2 < -10000)) return 'B'; //
```

```
      palm is closed --> finger and handback is in a line
      opposite direction.
209   else return '0';// not moving forward or backward
210 }
211
212 char LoR(int x1, int y1, int z1, int x2, int y2, int z2) {
213   if (y1 < -11000 || y1 > 5000) return '0'; // axis of hand
      won't be horizontal
214   else if (z1 >= 10000) return 'R'; // palm is upwards
215   else if (z1 <= -13000) return 'L'; // palm is downwards
216   else return '0';// not turning
217 }
218
219 char isaction(int x1, int y1, int z1, int x2, int y2, int z2
    ) {
220   if (UoD(x1, y1, z1, x2, y2, z2) == '0' &&
221       FoB(x1, y1, z1, x2, y2, z2) == '0' &&
222       LoR(x1, y1, z1, x2, y2, z2) == '0') return '0'; // not
          up or down or left or right or forward or backward.
223   else return '1';
224 }
```

Appendix B Raw Data from Accelerometers

Number	Gesture	Finger Signals			Hand Signals		
		x	y	z	x	y	z
1	Left	7234	-10896	7853	-1082	-3245	-7255
2	Left	8256	-12054	6598	-2036	-3940	-8937
3	Left	7264	-13515	6987	-468	-2315	-10246
4	Left	5976	-10579	9315	-2541	-3846	-8521
5	Left	6245	-9576	8316	-1851	-1957	-6048
6	Right	6480	-12756	8645	-2417	5261	8214
7	Right	7537	-11773	7459	-4237	4031	7275
8	Right	8450	-10786	6317	-1753	3567	6438
9	Right	9453	-9732	7059	-756	3795	7376
10	Right	6787	-11975	9374	-2756	4850	6177
11	Forward	-12688	-2908	3962	-15363	-5284	4479
12	Forward	-13209	-3154	4956	-13064	-8246	5976
13	Forward	-11753	-4962	5752	-12753	-6453	5737
14	Forward	-10753	-1753	3756	-15735	-7537	6752
15	Forward	-14775	-2753	4576	-13057	-6554	7753
16	Backward	-12092	-14592	15228	-12753	-8673	5372
17	Backward	-12737	-12372	13576	-9735	-6037	4975
18	Backward	-13752	-11934	16153	-10573	-7537	5327
19	Backward	-10027	-10573	15376	-9435	-6725	3575
20	Backward	-13576	-13707	14835	-9145	-8274	6905
21	Up	4106	-3839	-11096	11765	3342	10903
22	Up	3861	-3546	-12209	10549	2497	11506
23	Up	2647	-4095	-11873	9576	3016	10946
24	Up	3672	-4846	-10946	9670	4946	9467
25	Up	4617	-4237	-9279	11064	3541	10940
26	Down	913	-13684	-11406	5412	8561	4252
27	Down	16	-15798	-12467	3164	7615	3446
28	Down	-715	-16549	-11705	4972	6498	5012
29	Down	467	-14679	-12649	5167	7164	4792
30	Down	-59	-13098	-12210	4902	8497	4843

Table 3: Raw data for gesture recognition