EE202C

Networked Embedded Systems Design

Smart Drone

Design Document

Peidong Chen, Jiayu Guo, Yitian Hu, Yang Yang

**Version 1**

**Date 10/14/15**

Revision History

|  |  |  |  |
| --- | --- | --- | --- |
| **Version** | **Date** | **Author** | **Comments** |
| Version No | Date Completed | Author(s) | Comments on Versions |
| 1 | 10/10/15 | Team | Initial Version |
|  |  |  |  |
|  |  |  |  |

Table of Contents

Revision History 2

Table of Contents 3

1 Abstract 4

2 Team 4

3 Technical Approach 4

4 System Design 4

5 Implementation Schedule 4

6 Implementation Description 5

7 System Source Code 5

8 Test and Evaluation System 5

9 Test System Source Code Repository 5

10 Test Results 5

11 Analysis 6

12 Shared Systems 6

13 References 6

# Abstract

Our goal is to design a robust smart drone on Intel Edison IOT platform from a fundamental point and further elaborate its functionality to photography, goods delivery, multi-drone communication. It is a system integrated with Edison microprocessor, a couple of sensors and four propellers. There are several challenging but critical issues involved in this project. First, the assembly of our drone requires precise installation of motors, battery, sensor and Edison board, so that the center of gravity is exactly in the geometrical center of the drone and the whole structure has the minimal weight. Second, sensor dark noise, which directly degrades our signal quality and affects the drone flight control loop, is another key factor. Both analog filters and digital filters are necessary for our system calibration. Third, flying control algorithm, which is based on sensor data, is also critical for drone balance, movement and landing when lack of battery. Our system has the ability to deal with real-time control signal and convey it accurately through our server and SSH remote connection. Fourth, the command sending and data collecting function makes sure that our drone has an instantaneous reaction through other IOTs and the ability of communication on cloud.

# Team

***Yang Yang:*** On air IOS application (used to get and send GPS information, taking pictures) and off-air remote control/test IOS (used to send control or test signal) application design; Smart drone state (according to sensor information) analysis and motor control signal output module implementation; Evadible system design; Multi-Smart Drone following Algorithm.

***Peidong Chen:*** Server side APi design (using to send control commands, drone status and all the data collected); Server side database build (used to store the data collected); Smart drone state analysis (according to sensor information) and to motor control signal output module implementation; Evadible system design; Multi-Smart Drone following Algorithm.

***Yitian Hu:*** Data acquisition and analysis from MPU9025 9-asix sensor module; learn and apply AHRS sensor fusion algorithm based on 9-asix data from MPU9250. Smart drone state analysis (according to sensor information) and motor control state machine algorithm design; Drone assembly.

***Jiayu Guo:*** Algorithm programming for flight loop control; Sensor signal processing and noise calibration; Test the functionality of drone system; Ultrasonic module for distance detection.

# Technical Approach

For this section please include:

1. One page description
2. This includes a discussion of general objectives
3. This may also includes a brief discussion of known or anticipated limitations (that may be entirely justifiable).

Our objective is to design a smart drone based on Intel Edison IOT platform. As a smart drone, it should be able to fly autonomously and communicate with another smart drone to complete the task like taking pictures, following and detection. The drone has the capability to navigate using GPS/Wifi and to arrive at a specific destination. Furthermore, functionality as an anti-drone drone, which control some anonymous drones into restricted area, and transformer-drone, drone can intelligently change its shape could also be extended.

To be specific, it is fundamental for this smart drone to stay on air and fly stably. It should fulfill the function like moving forward, backward, changing direction and altitude by control the speed of each motor. The flight control loop would autonomously balance the position of the UAV with the aid of accelerometer and gyroscope. Then, the smart drone should be able to track its location by using its carried iPhone and fly automatically to a specific place according to the received GPS information. In this task, it is necessary to implement the evadible function to help the drone fly safely without any collision into obstacles. One key limitation is the signal accuracy from our GPS module. After this, we will implement another smart drone to realize function between two chips of Intel Edison. The second smart drone should, firstly, have the function of following. In this model, we can send a movement command to one smart drone and the other one will follow the former one. Secondly, we will design a detecting algorithm for the smart drone to detecting the other one. It should be able to distinguish smart drone from obstacle, birds and some other objects. As an anti-drone drone, the second drone would recognize any invaders immediately. It has the features like taking photos of these invaders, following and preventing invasion and collaboration with other anti-drone drones.

However, there are also some anticipated limitations. First, the commercial GPS accuracy is around 10m, which will increase the error in navigation process. A 10cm level resolution is preferred as our location system. Also, the signal is attenuated aggressively when the drone is indoor. Second, our battery life time is usually around 15-20min for current technology. To improve our drone' s cruise time, some remedies would be tested to realize a drone would do self-battery-change or self-charge. Third, although a bunch of calibration methods can be included to minimize the noise from our sensor, the reliability of the system can be challenged by any signal delay during the data-path. Last but not the least, data interaction between the Edison and our server would lead to the data security issue and sampling time consideration from our microprocessor. Thus, our drone may not really have the real-time interaction with all the surrounding environment information.

# System Design

For this section please include:

1. One page description of implementation
   * Include description of algorithms
   * Include architectural diagrams
   * Include discussion of specific challenges or missing elements that must be acquired.



Fig.1 Architectural Diagrams

The architectural diagrams are shown above. There are many challenges in our drone project because we have to build the drone from the very beginning. We need to assemble the drone and pass the strict balance requirement. Since we do not have a remoter, we need to find a way to control our drone remotely using our own cell phones. Then we need to choose the accuracy of our sensor system and the latency of our processing system so that the drone can avoid vibration during flight. As for how to reduce the sensor noise and design suitable movement and balance algorithms, a great amount of efforts need to be made. We will explain them more specifically as follows.

First, as for the assembly of our drone, we will make sure that after all the parts are installed on our drone, the center of gravity lays on the geometrical center of our drone. Or in other words, the center of gravity should be the intersection of two pairs of opposite motors, so that when the drone suspends, the RPM values of all the motors will be the same. It will help simplify our motor control.

Second, the drone is supposed to be controlled by smart phones remotely, so we cannot use phone to control using bluetooth for the simple reason that bluetooth has a range limit. Also, we cannot use "ssh method" to connect the drone, since the drone doesn't have a public ip address or public domain. Then we use a server "fryer.ee.ucla.edu" to be the solution. When we want to control the drone, the iPhone sends the command to the server, and the drone fetches the command from the server. Then the drone will react as the command does. Also, the drone can send its data to the server, so that we can get it and save it for analyzing purposes. The processing is shown below.

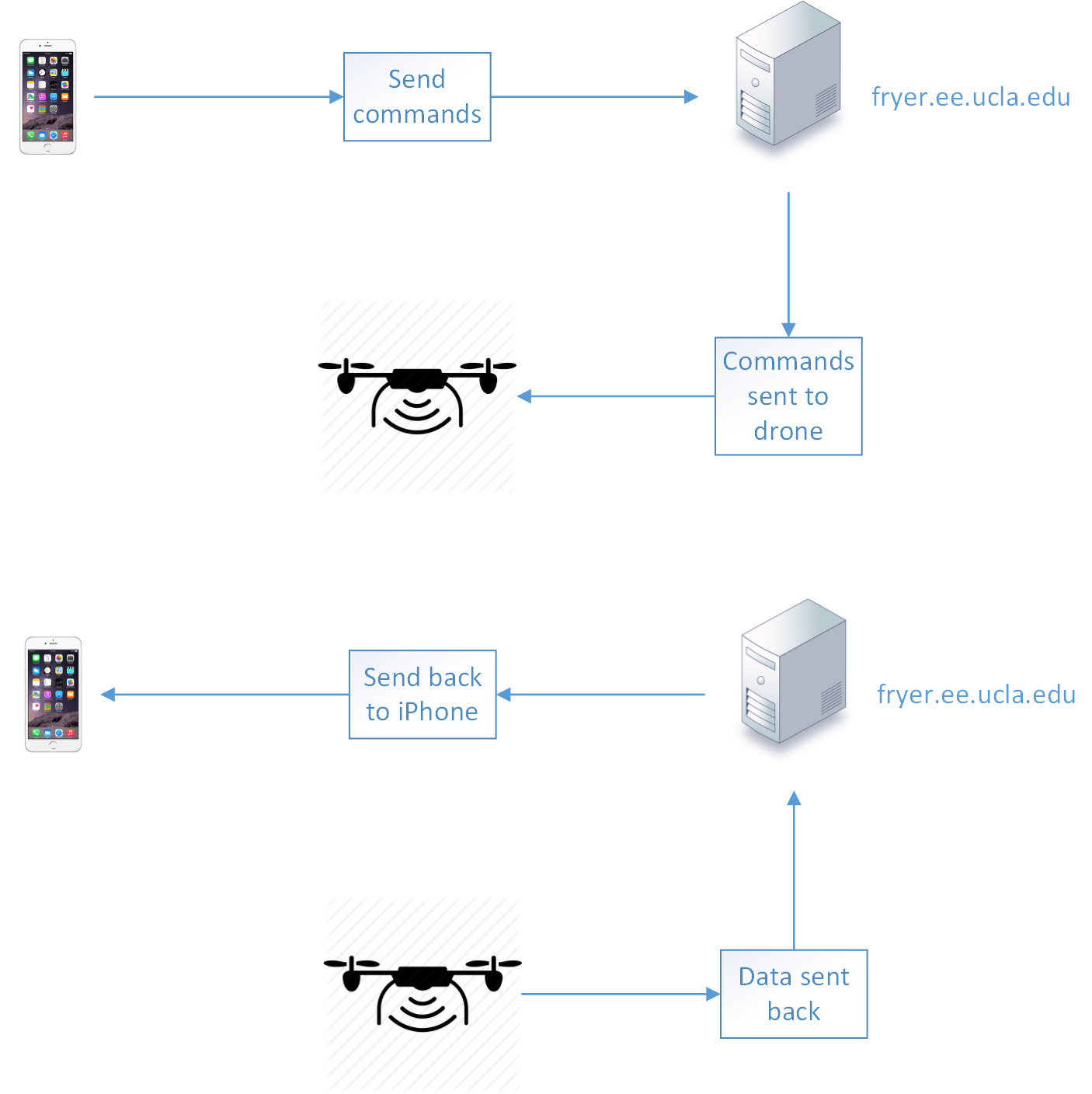


Fig.2 Data transfer processing

Third, we may add a function called "flight learning", which means that when the drone changes weight, we can start the "flight learning" function. Then the drone can learn the PWM variable value, at which the drone can suspend in air. In general, the drone has 10 actions: up, down, right, left, forward, backward, clockwise rotating, anticlockwise rotating, suspending, stop. All other nine actions are based on the suspending action, which means that when we get the value of suspending action, we can do a small amendment to do all other nine actions.

Consider the following image, the four propellers are called propeller1, propeller2, propeller3 and propeller4. And the PWM duty cycle are named P1, P2, P3 and P4.

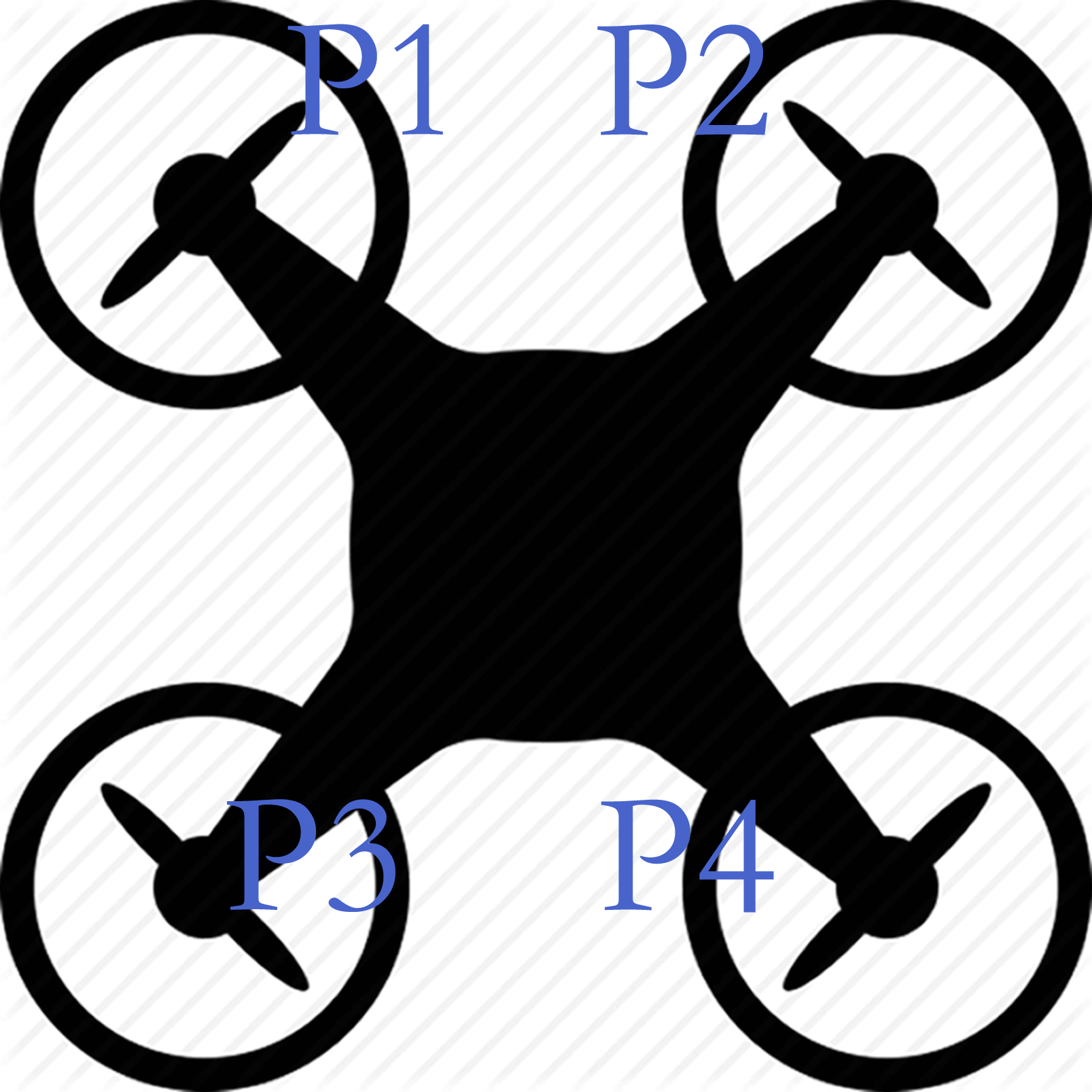


Fig.3 Demonstration of drone propellers

When the drone is suspending, P1 = P2 = s1 and P3 = P4 = s2. (Supposing P1 and P4 are clockwise, P2 and P3 are anticlockwise. And this image is in Bird's-eye view.)

1. Flying up action : P1 = P2 = s1 + a, P3 = P4 = s2 + a

2. Flying down action: P1 = P2 = s1 - a, P3 = P4 = s2 - a

3. Flying right action: P1 = s1 + a, P2 = s1, P3 = s2 + a, P4 = s2

4. Flying left action: P1 = s1, P2 = s1+a, P3 = s2, P4 = s2+a

5. Flying forward action: P1 = s1, P2 = s1, P3 = s2 + a, P4 = s2 + a

6. Flying backward action: P1 = s1 + a, P2 = s1 + a, P3 = s2, P4 = s2

7. Clockwise rotating action: P1 = s1, P2 = s1 + a, P3 = s2, P4 = s2 + a

8. Anticlockwise rotating action: P1 = s1 +a, P2 = s1, P3 = s2 + a, P4 = s2

Fourth, as for the sensor noise, filters should be applied. In the MPU9250 chip, we have 10 options of digital low filter bandwidth, among which we may choose the lowest value, 5 Hz. However we will suffer from 33.48ms delay as a trade-off. Also, if the noise still influence the performance of the drone badly, we may try some filter algorithms such as Kalman filter, which uses a series of measurements observed over time, containing statistical noise and other inaccuracies, and produces estimates of unknown variables that tend to be more precise than those based on a single measurement alone.

Fifth, as for the so called "automatically avoiding obstacles" feature, we will install at least six ultrasound sensors. For example, if the drone goes straight on in a forward direction, the data of the ultrasound sensor which is placed in front of the drone will be monitored. If there is some obstacles in front of the drone, it will attempt to try other directions that does not have obstacles.

Sixth, when it comes to the latency, actually we worry a lot. Probably, the balance of drone need a very low latency, otherwise, it is very easy to encounter a case that before the last commend of adjustment is done, a new commend is sent to the processer. Then we need to reduce the complexity of our data process algorithm and motor control algorithm. We have to run some tests on our drone to know how critical the latency is.

Seventh, for the following function, the followed drone should be able to record the movement information like GPS, velocity, acceleration and direction. After that, it send these information to the following drone to let the following drone go after it. The challenging is that we should design an algorithm to compute the path so that the drone can be followed correctly. We come up an idea that we can install a certain pattern composed by points of infrared LEDs on the master drone, so that we can use infrared sensor on the slave drone to detect the position of the master one and implement the follow.

Eighth, we will add a model named patrol. The drone will patrol around a specific area. Once it detecting an invader, it will take pictures and report to us. For the detecting function, we are now planning to use a camera to take pictures automatically, then using image pattern recognition algorithm to detect the potential drone.

# Implementation Schedule

This is a critical step. For this section please include dates for:

1. Design completion date: 10/5/15
2. Test plan completion date: 10/12/15
3. Design review date of Midterm in Week 6 **to include demonstrations of system operation 11/4/15**
4. Implementation completion date: 11/20/15
5. Testing phase complete date: 12/01/15

# Implementation Description

For this section please include:

1. Two to five pages
2. This is an Implementation narrative description that should guide others who will learn about the architecture and approach you have used.
3. This augments the Section 2 and is primarily focused here on approach and the systems you used to compose your solution.
4. Please describe approaches used for implementation. This may include a description of the development of your system state machine, individual software modules, data acquisition methods, or others.

1. Drone components assembly part

Mainly, we will assembly the drone whenever we are ready for test. As we mentioned above, how and where to install all the components on our drone matters much for the performance of our drone. Supposing we have 4 exactly same propellers, it will be the best case if we can make all the four propellers rotate at the same speed during balance. That is because it will simply the flying auto-control algorithms. So during the assembly operation, we should allocate the weight symmetrically as much as we can. And after we install all the components, we will test if the gravity center is the geometry center by the following operations. First we fix two ends of two opposite wings, and we check whether the other two wings are balanced. Then we fix two ends of two other opposite wings and we check the balance. After the two steps, we can make sure that the drone is correctly assembled.

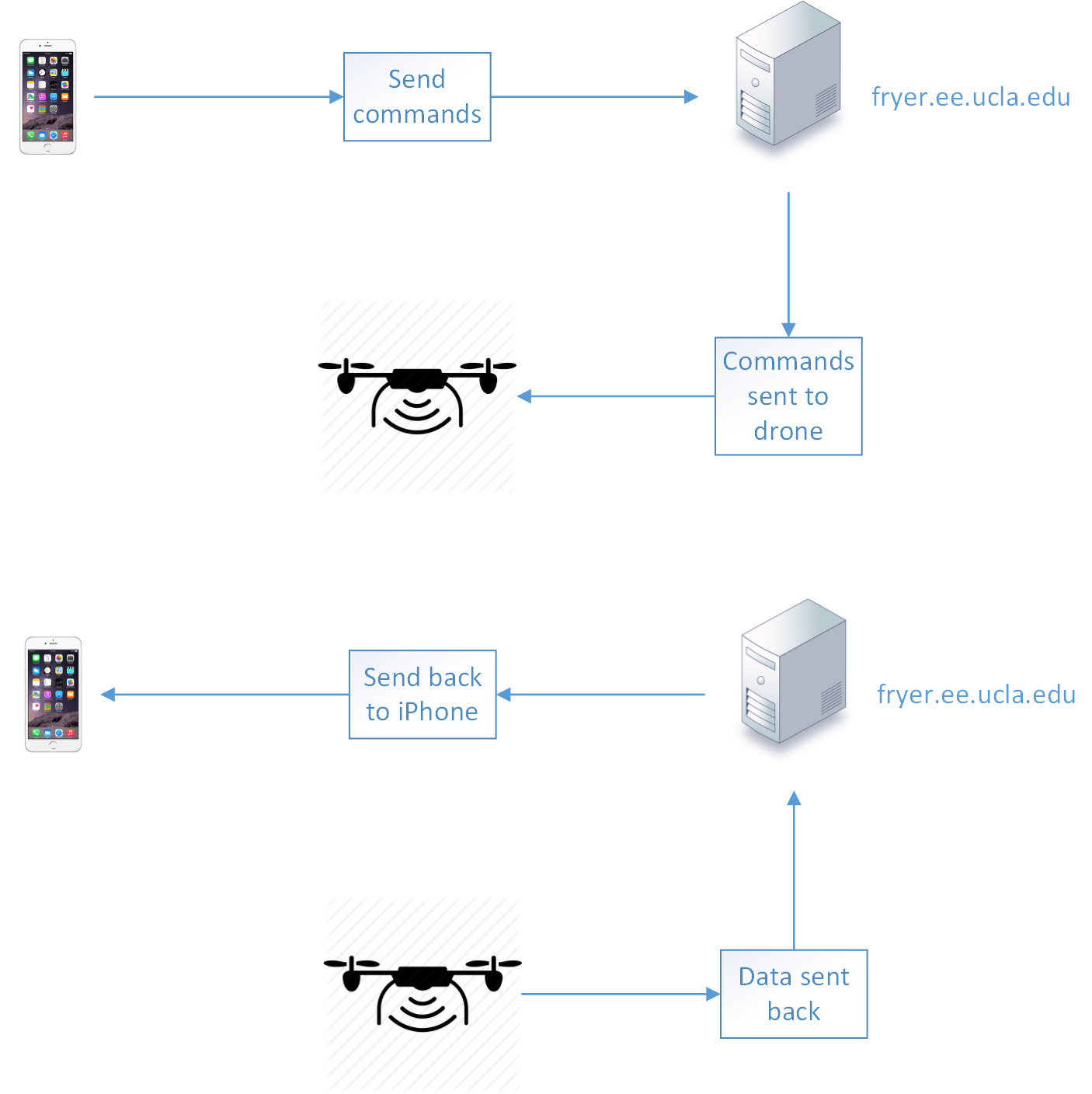
2. Sensor data processing part (ultrasound sensor, gyroscope sensor, accelerator sensor, magnetic sensor)

The sensor data processing part would be divided into two categories. The first category consists of gyroscope sensor and accelerator sensor would provide us real-time data for the drone's 3D information. The flight position is directly controlled inside this highly sensitive and high resolution feedback loop. The feedback loop prevents the drone from crashing in any potential risky conditions. To obtain accurate data result, one method is to alleviate the noise from our front sensor to the minimal level. A filter implemented by quaternion representation, allowing the magnetic distortion and gyroscope bias drift reduced to achieve best MARG sensor arrays performance. The second category would be the ultrasound sensor, which contains the distance information and monitors the surrounding environment. It is reasonable to say that these sensors could be interacted in a slower communication rate. But still there is the issue concerned with the detection angle between the sensor and the barrier. To build up the whole sensor arrays, all the sensors would be connected to the Edison board by I2c bus.

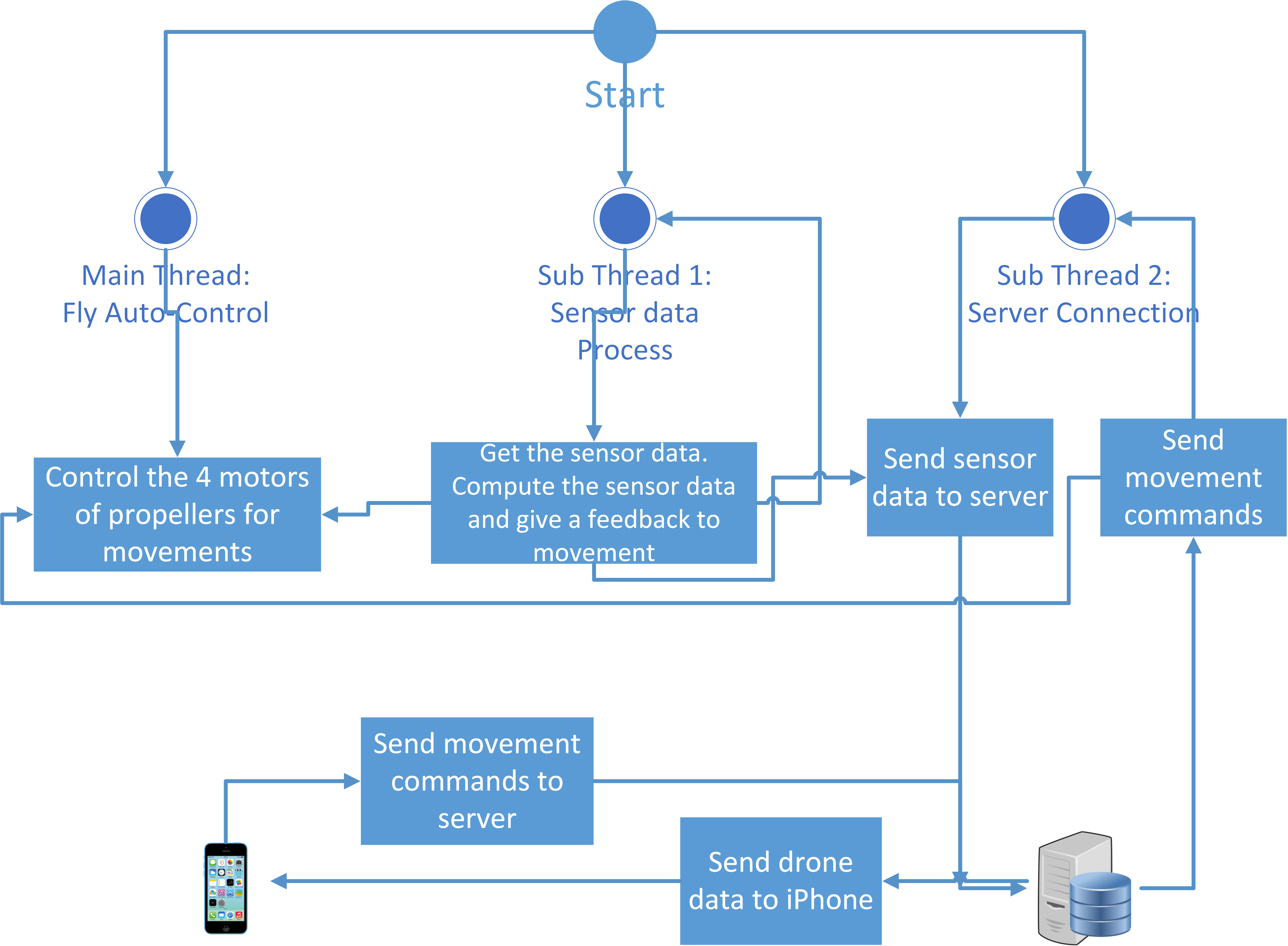
3. Motor control part

The four motors on the drone will be controlled by four ESC modules. The input of ESC is PWM signal. The duty cycle of the ESC input will modulate the frequency of the 3-phase AC current of a motor, which will then change the speed of rotation. As for the PWM generation, we will use the four PWM output pins which are 3, 5, 6 and 9.

3. Cloud server part



As shown in the figure above, we will need to let the iPhone communicate with the drone. Because both our iPhone and drone do not have public IP addresses, we need to use a server as a solution for the communication.



We use HTTP Restful request as a communication way between devices and the server.

Also we use MySQL as the database for managing the data that has been sent from the drones.

4. Mobile application part

4.1 On air application

The smartphone carried an iPhone to provide GPS information and map information so that it can navigation automatically. The iOS application is implemented by xcode 7. For the GPS information, we use Relocation/CoreLocation.h packages to get the information.And AFNetworking to send and get information from server. We also use this iPhone to take pictures and send it to server.

4.2 Test/control application

In this part, we use a slider to control the speed of each motor. We can change the value of sliders(including button maximum, minimum and medium), and send these value to server. Edison, using C code, can get these information from web. According to these values, it can then change the speed of motor. We also use AFNetworking to send information to server. Its just use as a remote control.

5. Flat flying control part

Based on the sensor fusion data we get after processing the raw data of the MPU9250, flat flying control algorithms can be designed. Since during the fly, any small tilt will accelerate the drone, the processed date should be very accurate and real-time, which means that the latency should be as small as possible. And in order to make sure that the drone fly flatly and steadily, we will basically simplify the flying into 9 modes: suspending, ascending, descending, going forward, going back, going right, going left, clockwise rotation and anticlockwise rotation. For going forward, back, right and left, we only need to accelerate two neighboring motors. Specifically, first we record the suspend rotation speed(R\_sus) and define the maximum tilt angle for each movement (e.g. 10°). Second, if the sensor tells that the drone reaches 10°, we adjust all the four motors' speed to R\_sus×(1+sin10°). And then the drone will not falling down because of a small tilt. For clockwise and anticlockwise rotation, we just need to speed up two opposite motors and speed down other two motors, the drone will rotate.

6. Obstacles avoidance part

This smart drone can fly autonomously. We use ultrasonic sensors to detect obstacles, the drone will steer automatically according to the information of ultrasonic sensors. When detecting a obstacle, the drone will, firstly, try to change the altitude to bypass the obstacle. If after changing a certain meter of altitude, the obstacle can still be detected, the drone will try to turn left and right. If obstacles can still be detected on the left and right side, the drone will go backward.

7. Drones follow flying part

In this part we plan to use GPS as the location method. We put two iPhones on two drones and these iPhones can set up hotspot using AT&T's network to connect drones to the Internet. And also we can utilize the iPhone's GPS data to locate the drone's location. In the outer space, the drone's GPS accuracy is good enough to do the drones' follow flying function.

# System Source Code

# Test and Evaluation System

1. This description of the design of your test and evaluation system developed to test the design hypotheses described above.

# Test System Source Code Repository

1. This includes source code directory paths
2. These test systems may be shell scripts, for example that demonstrate your system and determine its response.

# Test Results

1. This is flexible. Please use as many pages as required here.
2. This will include both text and graphics
3. This section should include data describing test results for
   1. System performance with both success and failure displayed
   2. System operation (for example showing a sequence of operations described in a table form.

# Analysis

1. This is flexible. Please use as many pages as required here.
2. This is an opportunity to highlight the success of your development. Please note that there will be discoveries that show performance better than expectations and less than expectations. It is important that we describe both and give reasons. Of course, our objective is to be successful, but ,we recognize that there will be shortfalls. Experience in development helps us reduce the frequency and severity of shortfalls in design.
3. This section provides a discussion of results including:
   1. Comparison of actual operation with the operation expected at the time of design, at the midterm.
   2. Discussion of this performance describing reasons why performance may exceed or not meet objectives.
   3. Discussion of regions of applicability for our designs. That is, this section should describe regions where performance meets objectives and those where it does not.

# Shared Systems

1. Please list systems that you would wish to share with others for future use in future courses, or also in research. These systems could be used in their entirety or could inspire other utilities that are useful. Future courses will acknowledge your contributions.
2. Please provide a path to specific source code modules, for example.

# References

1. Please list references from publications or web sources that have been useful in your development.