Team Daphne: Amphibian Quadcopter

Abstract

Nowadays, drones can only operate in the one dimensional space of the sky. It makes a lot of places inaccessible due to the fact that most drones aren't waterproof or not dexterous underwater. Our team wants to break the boundary of the operational spaces of the drone. In this sense, we can greatly extend the possible applications of drones, such as geological surveys or military use. Through the preparations of last quarter and the implementations of the spring quarter. Our amphibian quadcopter is be able to launch in air and conduct underwater controls through wireless signals. The PID controller of the quadcopter functions correctly and optimize the controls of quadcopter in air and underwater. All electronics are waterproof-coated and free of damage underwater and control signals is able to transmit to the quadcopter underwater at a maximum of 20-centimeter deep.

Introduction

Throughout the course of this quarter and the preparations of last quarter. Our team has studied topics of Digital Motion Processing (DMP), using Pulse Width Modulation (PWM) for motor control, system and control design, common electronic hardwares and wireless communications. We designed and achieved essential realizations of the amphibian drone throughout these two quarter. Inspired by the previous study of the underwater drone designed by a team at Rutgers University in New Jersey, our team wish to produce prototype of an cheap, accessible drone for scientific research, photography and military purposes. In the following sections, we will be discussing the topics of control theories, electronic hardware, waterproof, how we were able to achieve our goals of producing a prototype within short period of time and the end result of our works . For the implementation of the hardware and software of the system, our team utilized several open-source libraries and contributions from the UCLA-IEEE members. Their works are cited and referenced in the last pages.

Background

The project of amphibian drone is built upon the following electrical engineering foundations, two of which we relied on heavily are the basic knowledges of DMP and the concept of PWM controls.

Digital Motion Processor (DMP)

The MPU6050 is a 6-DOF IMU sensor that consists of an accelerometer and a gyroscope. The accelerometer gives the acceleration at three axes, and the gyroscope measures the angular velocity with respect to x, y and z axes. These six values provide valuable information about the motion of the subject, and we can use the accelerometer values to calculate the orientation variables, such as roll, pitch and yaw.
However, raw values read from the IMU are very noisy and we need to perform sensor fusion using extended kalman filter to calculate the euler angles without much noise. It brings a lot of additional work, but fortunately, the IMU itself has a built-in component called Digital Motion Processor that fuses the raw data from accelerometer and gyroscope and calculate the orientation variables with much less noise. The DMP component is not documented officially, but there is an Arduino library developed by Jeff Rowberg on DMP implementation that is well-rounded and easy to use.

PWM

Pulse-Width Modulation (PWM) is a modulation technique widely used in power control. A switch between the supply and load (usually a MOSFET) is turned on and off at a very fast rate, and the average voltage fed to the load is controlled by duty cycle, which is the percentage of one period in which a signal or system is active. When the duty cycle increases, a higher equivalent DC voltage will be applied to the load. The duty cycle can be easily controlled by the user, which makes PWM the most convenient way to generate varying DC voltages.

Methods

System Overview

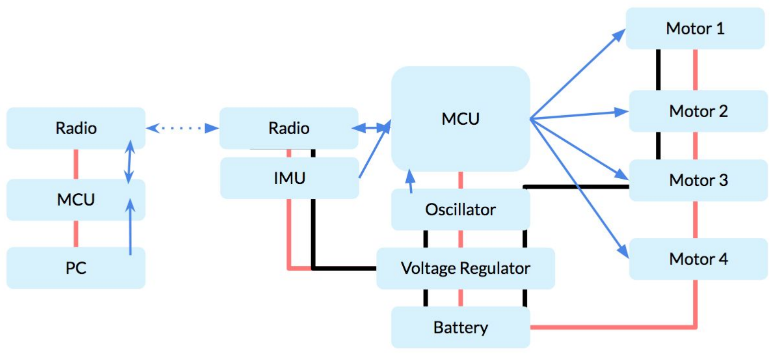

Fig. 1. Block diagram of whole system

The figure above shows an overview of the whole system. On the left we have the controller side, and it communicate with the radio on quadcopter through 2.4GHz WiFi (indicated by dashed line). The onboard MCU of quadcopter take in measurement from IMU and calculate the corresponding PWM outputs and pass them to four motors. In the case of powering, maximum of 7.4V power is directly applied the motors and 3.3V regulated power is used for powering the IMU and MCU.

Parts Choice

In the following section, we will discuss all the hardware components we used in the project, including why we choose certain parts, their performance and functionalities.

Frame

There are several 3d-printed micro quadcopter frames available online for free, we choose the one from https://www.thingiverse.com/thing:1221911. Because it is robust in design and lightweight(around 40g including board and motors).

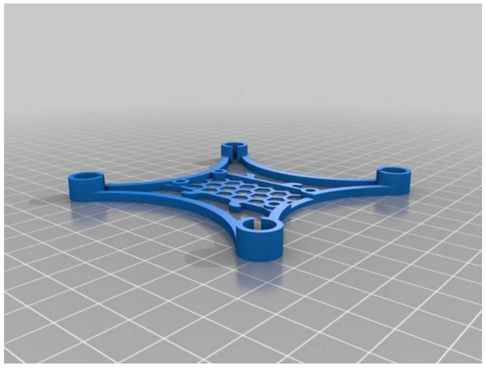

Fig. 2. CAD model of quadcopter frame

Motors

Four 8.5mm brushless motor are used for our quadcopter, according to the specification of 8.5mm motor it can produce up to 40g thrust when driven at around 3.7V, which is more than enough to lift our circuit board and quadcopter frame up in the air. In addition, brushless motor are usually appeared to be waterproofed. We tested those motors underwater and they works perfectly fine. We choose 8.5mm motor because it is compatible with our frame and providing enough thrust for our system. Furthermore, larger motors tend to drain battery faster so will results in increasing total weight due to changing to larger battery.

WiFi Module

WiFi Module is used to implement the communication between controller and our quadcopter. The model nRF24L01, a single-chip 2.4GHz WiFi Module, is chosen because this model is one of the most commonly used model for embedded system design and we had some previous experience with this module. nRF24L01 operates in 2.4GHz ISM band, allowing three different transmission rate 250kbps, 1Mbps and 2Mbps on air.

Furthermore , it has very low power consumption and operates using 3.3V regulated power so that we won't waste too much battery life in communication.

IMU (Inertial Measurement Unit)

In order to get the all the six desired states that we need for controlling this system, MPU 6050 chip is chosen as our IMU. MPU 6050 is a six-axis gyro and accelerometer so that it can provide acceleration and angular velocity data(total 6-DOF). This chip requires power ranges from 2.375V-3.46V, thus regulated 3.3V power is supplied to it. In addition, we have used MPU in one of the lab during EE183DA, it has well-documented Arduino libraries online so make it easier for us to integrate it in our system. One thing to be notice is that MPU also contains a built-in DMP (Digital Motion Processing Unit), meaning all the sensor fusion has been done in the given code. We can easily get yaw, pitch, roll and acceleration from the IMU.

Microcontroller Unit(MCU)

ATmega 328P is chosen as our microcontroller for this system. ATmega 328P is the model used in Arduino UNO, so it guarantee that it has sufficient computation power for processing data and calculate control outputs. Since we have four motors on the quadcopter, at least 4 PWM pins are required for our MCU, ATmeaga 328P also meets this requirement. In addition, we choose to use solely the MCU chip instead of an Arduino board because we want to reduce size and weight of the circuit board as much as possible. As a result, we need add additional components(Oscillator, Voltage Regulator and FTDI bootloader) along side with the MCU chip to make it functional, those parts will be discussed in the following paragraphs.

Oscillator

The MCU needs an external clock to keep track of time, which is provided by the oscillator. In this case, we chose a 8MHz Ceramic Resonator. It is cheap and relatively easy to assemble on board , and its precision is good enough for MCU.

Voltage Regulator

Voltage regulator MIC5219-3.3YM5-TR is chosen for our circuit. This voltage regulator can take in input up to 12V and output 3.3V, which is compatible with the battery voltage(7.4V). It also has good dropout voltage and current capability.

Battery

We choose Turnigy 500mAh 2S Lipo Battery for our quadcopter, it has 2 cells(7.4V) for providing enough voltage for motors and MCU.We decide on Lipo battery because Lipo battery has better capacity/weight ratio than other types of batteries. 500mAh is chosen due to larger battery capacity will results in heavier battery.

Schematics & PCB Design

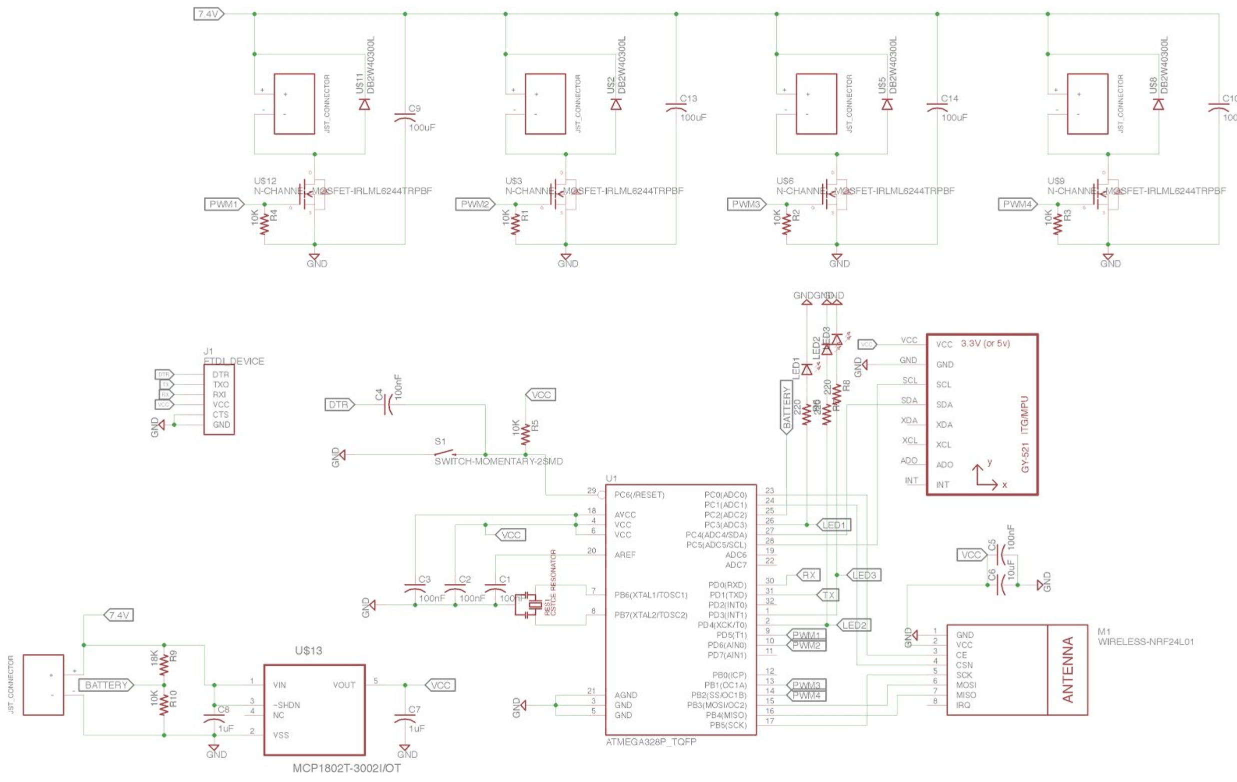

Fig. 3. Schematics of the circuit

The figure above shows the schematics of the circuit. On the top shows four low-side switching motor drive, which means the motor speeds are controlled by PWM output to the MOSFETs. On the bottom left we have the battery and voltage regulator part, the voltage divider circuit is there for battery status monitoring, and for voltage regulator we just connected it as instructed in the datasheet. In the middle we have the MCU, it is connected to IMU(upper right), bootloader(upper left) , oscillator (middle left) and WiFi module (bottom right). In addition, we connect several LEDs to MCU for testing purpose. Other than just connecting pins together, we add two capacitors for WiFi module to prevent sudden voltage drop. One more thing to be noticed is that only certain pins of ATmega 328p can output PWM signal, we have to choose four of those PWM pins for motor driver circuit. Additional capacitors in the circuit serves the purpose of either preventing voltage drop or filtering high frequency noise.

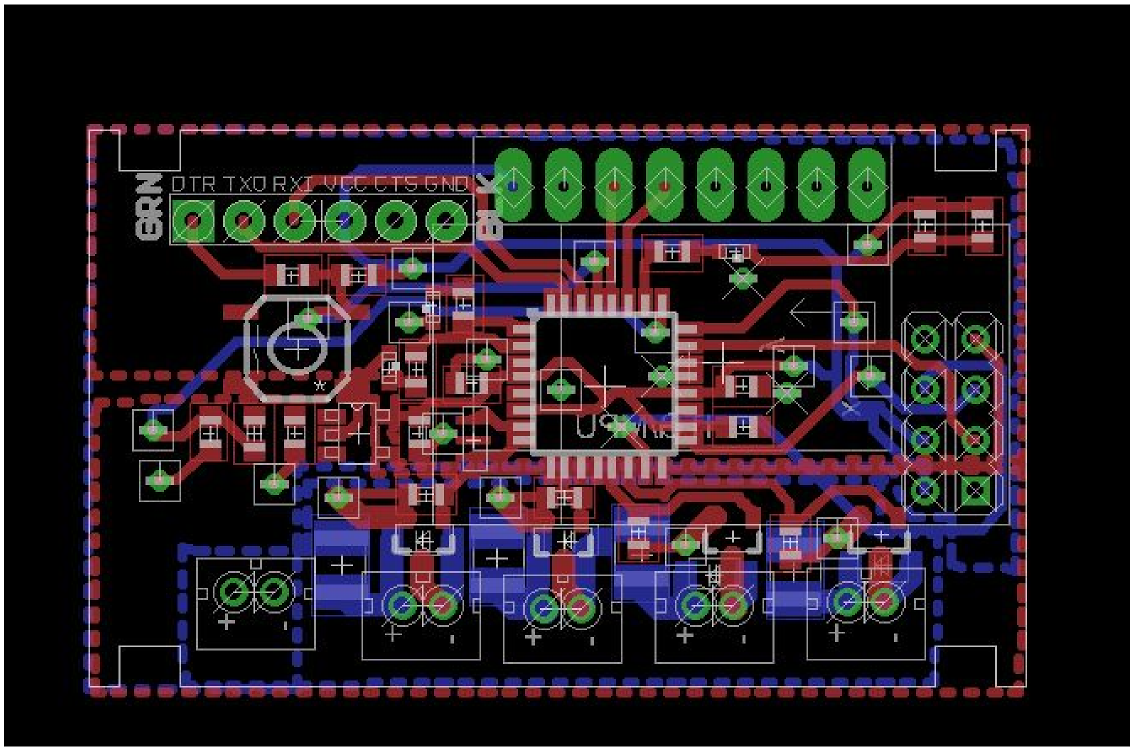

Fig. 4. PCB design of the circuit

The PCB board has dimension of 4.7cm by 2.8cm to fit onto the frame, which leaves us very limited space for placing components and routing. We decide on a two layers design meaning we place components on both sides of the board and connecting them through vias. Tracing widths are determined by using online PCB trace width calculator. http://www.4pcb.com/trace-width-calculator On our board, 0.4064mm trace width is used for most routing expect motors. Since motors are drawing current up to 4A, we use trace width 1.016mm.

Hardware Assembling

Motors and propellers

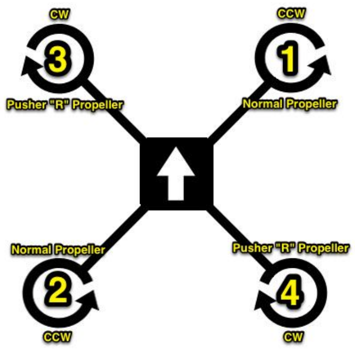

Fig. 5. Propellers and motors placement

For quadcopter we have two kinds of motors(clockwise and counterclockwise), and also two kinds of compatible propellers(normal and "R"). Motors and propellers have to be installed in the way that shown in the figure above to ensure we are able to control the quadcopter.

PCB Assembly and SMD soldering

For assembling the PCB we have to solder parts in certain order, oscillator first, voltage regulator second and MCU third. Oscillator is assembled by using heat gun and solder paste because the pins for oscillator are under the components so we cannot use solder iron to solder it on. In addition, we solder it first because heat gun will heat up all the soldered parts. After oscillator has been soldered, all other components can be soldered using solder iron.

Controller

We use a pre-built controller designed by UCLA IEEE officers Cullen Quine & Jeffrey Chen. The controller use the same WiFi module for communicating with the quadcopter. We are able to change PID parameters remotely and set WiFi channel number remotely. Furthermore, we can trim the roll, pitch and yaw set point in the controller for calibration.

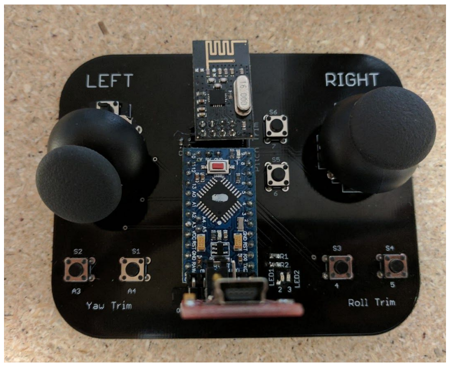

Fig. 6. Picture of Controller

Waterproofing

There are three stages of waterproof procedure we did for the hardware. First of all, we tested the claimed waterproof motor of our choice once we received them. For a simple testing, we dipped the motor directly into a cup of water with its wires attached as shown in the figure below.

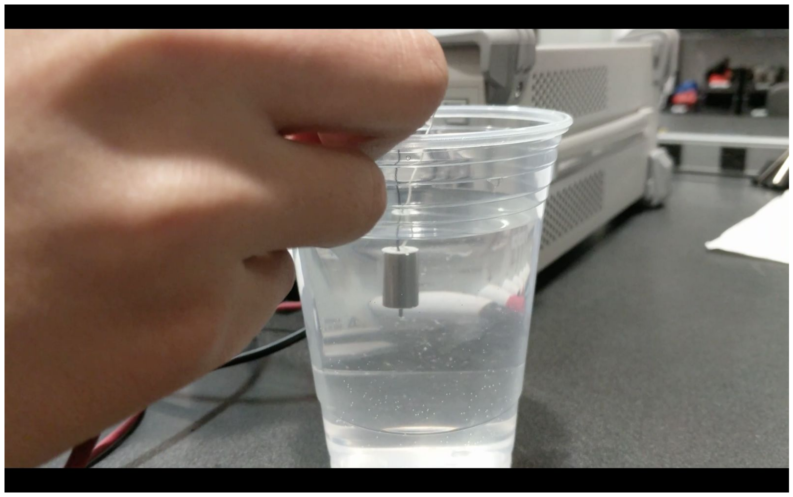

Fig. 7. Waterproof motor testing

Next, we found a technique called conformal coating, which is widely used in the military in harsh environments. The benefits of conformal coating are that it is PH-neutral, meaning it is free of erosion like common coating materials such as silicone. For delicate components like the PCB boards, it is important to consider the effects of acidic erosion for the sensitive surface.

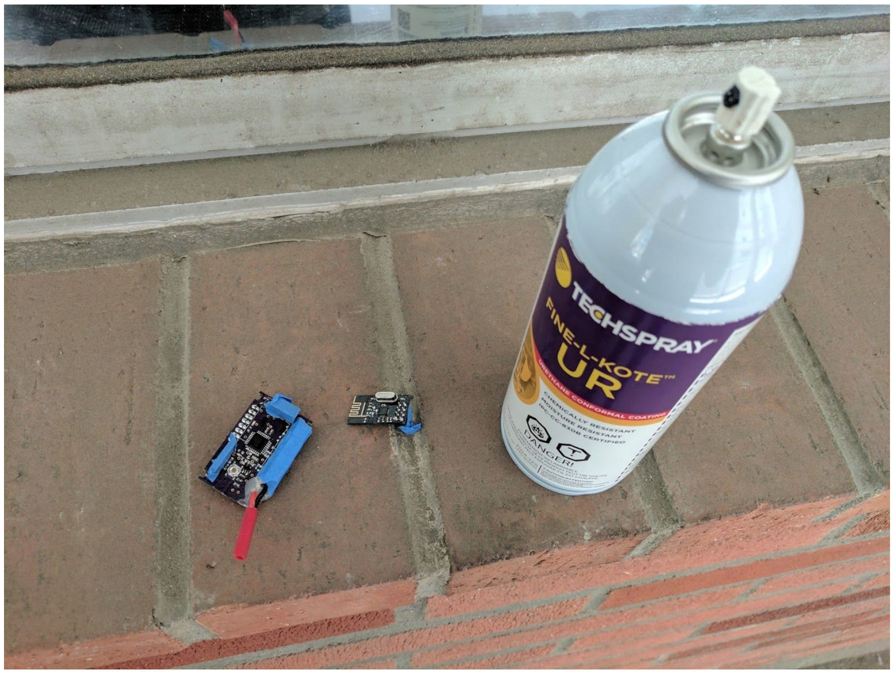

Fig. 8. Conformal coating of the PCB board

Lastly, for the conjunctions of wires and all other areas of the drone that needs to be insulated, we chose hot glue to wrap the conducting parts inside. We finished this step at the end, since we have to assemble all the parts and finish debugging of the system before we can finally seal everything in an irreversible manner.

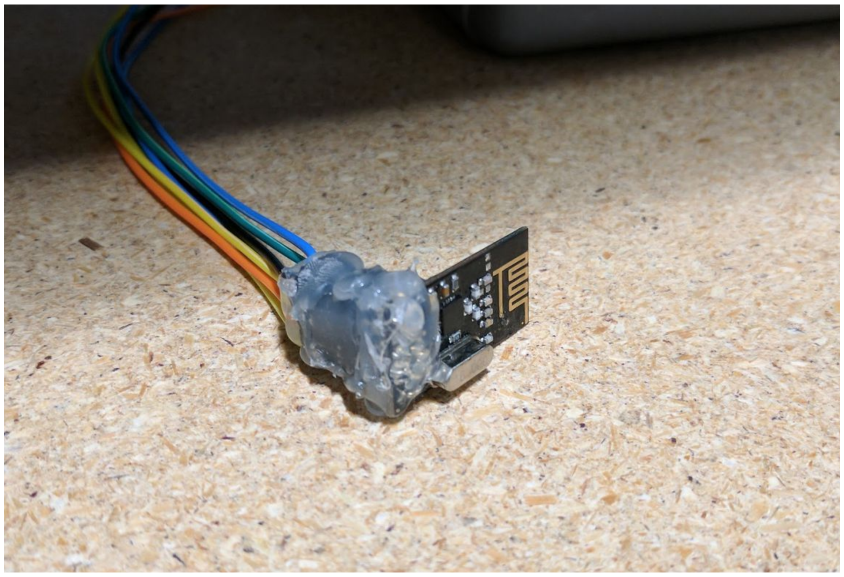

Fig. 9. Hot glue insulation for wire conjunctions

Overall, the results of waterproofing is satisfying, as we were able to conduct multiple testing successfully under the water. The conformal coating does run a risk of insulating the connection between wires and electric parts. As we found out, this is the only drawback of the spray painting process.

Control

Overall the quadcopter uses a feedback control system. The current state including roll, pitch and yaw rate of the quadcopter is obtained by the IMU, and the reference setpoint is determined by the controller input. After receiving the process variables, the MCU compares the current value with the setpoint, does PID calculation, and sends control signals to the actuator, which are PWM signals in this case. The PWM signals then drive the motors and the sensor values of the quadcopter change accordingly.

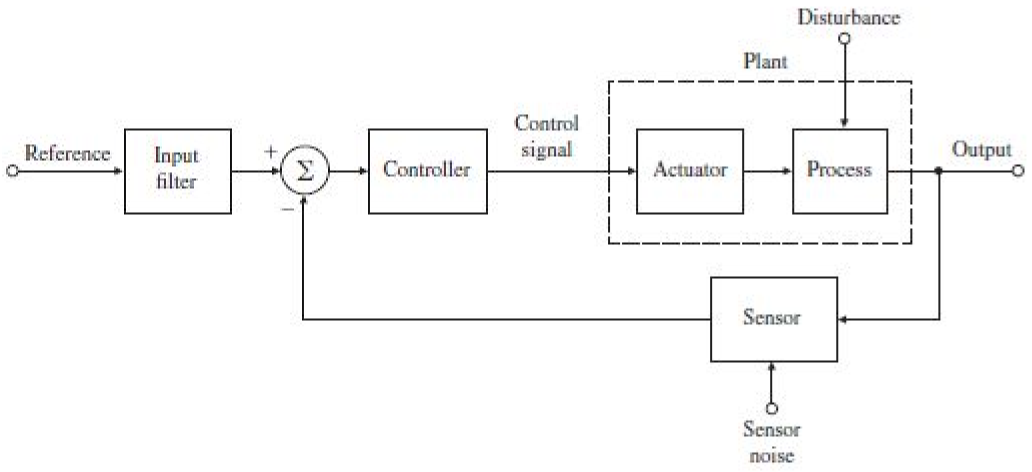

Fig. 10. Block diagram of an elementary feedback control

The specific control flow of quadcopter can be described as follows: to start with, we initialize the whole system, which includes initializing the radio and DMP function of IMU. After initialization, the system should wait in idle state. When in idle state, the quadcopter is still receiving values from the controller, but it will not drive the motors. The design of idle state is mostly because of safety concern: we'd like to turn on and off the quadcopter whenever we feel suitable.

In order to get the system out of idle state and vice versa, a special set of signals needs to be sent from the controller. The signals need to be unique so they won't be accidently sent during normal operation. Furthermore, for safety reason, the throttle component of controller values should be zero. In this case, the quadcopter is turned on when it receives maximum pitch and zero throttle, and it turns off with minimum pitch and zero throttle. It is extremely unlikely to send these signals during normal operation, so they can be used as unique gestures to toggle on and off the active state. Other combinations work as well, as long as they are unique and easy to memorize.

When the quadcopter is in active state, the MCU will run the control loop code over and over until it gets out of the active state. First, it will receive the controller values through WiFi module; the controller values include the throttle level, desired pitch and roll value, and PID coefficients. Then, it will read the current orientation from the IMU, including roll, pitch and yaw rate. The pitch and roll values from the IMU DMP have the range from -90 to 90. Specifically, the pitch increases when the quadcopter is moved forward, and it decreases when the quadcopter is tilted backwards; the roll decreases when the quadcopter tilts left, and increases when the quadcopter tilts right. Yaw rate can also be read from the gyroscope: it represents the rotation of the quadcopter with respect to z axis. It is positive when the system is rotating clockwise, and negative when it is rotating counterclockwise.

After that, we can perform the PID computations. The setpoints for roll and pitch are determined by the controller, and the setpoint for yaw rate is always zero because we don't want the quadcopter to spin horizontally. Ideally, there should be three sets of PID coefficients; however, the controller can only send one set of PID coefficients at a time. Therefore, we use the same PID coefficients received from controller for roll and pitch PID control, and another preset PID coefficient for yaw rate (relatively less important). After PID computation, we have four control signals: throttle, pitch, roll and yaw rate. We need to update the speed of the motor based on these signals. However, each motor has a different combination of these four signals in terms of summation and abstraction. To determine the specific +/- usage, we need to refer to the quadcopter dynamics which is illustrated below.

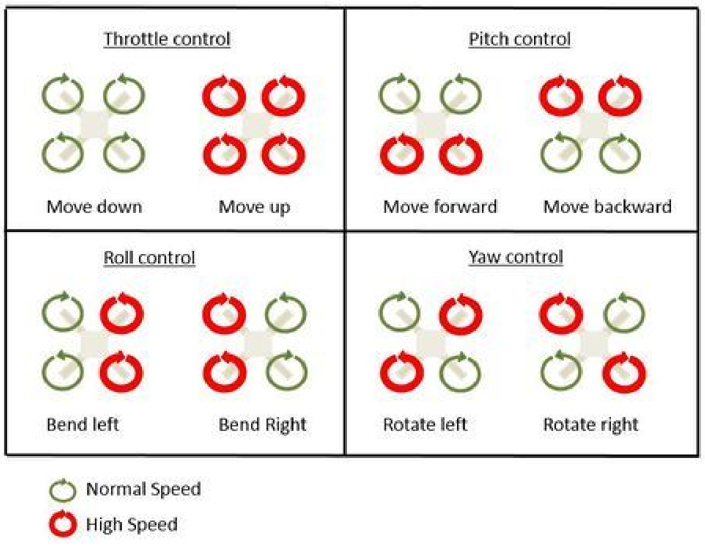

Fig. 11. Quadcopter dynamics

This image shows how the quadcopter can perform certain actions based on motor rotation speed. For example, all four motors need to be at high speed to move up, two motors at the back need to be at high speed to move forward (increase pitch), etc. Combine this with the previous info about roll, pitch and yaw rate, we are able to figure out the +/- usage for all for motors:

```
pwm_fl = throttle + pitch - roll - yaw;
pwm_fr = throttle + pitch + roll + yaw;
pwm_rl = throttle - pitch - roll + yaw;
pwm_rr  = throttle - pitch + roll - yaw;
```

Note that this is the +/- usage based on the placement and sign convention of the IMU in our design. When the design is different, these equations will also change.

Now we have the pwm values to write to the motors using analogWrite() function in Arduino. However, the pwm value indicating duty cycle is between 0 and 255, so we need to set a lower and an upper threshold. The lower threshold is 0; the upper threshold can be varied according to the demand. In this case, because we are using a 2S LiPo battery with maximum voltage of 8.4V to power the motors with maximum operation voltage 3.7V, the maximum upper threshold value should be (3.7)/(8.4)*255 = 112. The threshold that we ended up using is 70 during tests, although it can be increased to generate stronger performances from the motors.

Another important thing to do in the active state is to update the battery voltage. When 2S LiPo battery is fully charged it has output voltage of 8.4V. If the battery voltage drops below 7.4V it should be charged as soon as possible; when its voltage is below 6V (the minimum safe charge), it can't be safely charged anymore. Therefore, it's important to keep track of battery voltage. When the battery voltage is below a certain threshold, we need to kill the motors and turn off the quadcopter. The battery voltage is read by a simple voltage divider: connect a 18kΩ and a 10kΩ resistor in series between the battery and ground, and the voltage across the 10kΩ resistor when the battery is fully charged is 8.4*10/(10+18) = 3V, which can be read into the MCU through the analogRead() function. When the battery voltage reading is less than 7.4V, an LED on the controller side will turn on indicating low battery, and the quadcopter will enter idle state.

That is the ideal implementation of a battery voltage tracker. However, it turned out to be more complicated during the test. Because the motors draw a large current from the battery when they start spinning, the voltage output will decrease for a brief period of time. The algorithm described above cannot distinguish between this voltage dip and actual voltage decrease, so the quadcopter can't lift off even if the battery is fully charged. To solve this problem, we need to measure the elapsed time that the battery voltage is below the threshold. If this period is longer than a certain time (1s for example), then the MCU will turn on the LED and turn off the quadcopter.

Results

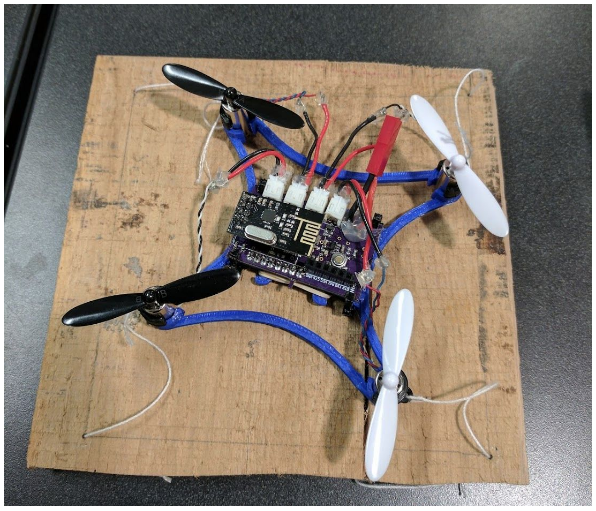
Fig. 12. Fully assembled quadcopter

Current prototype

The figure above shows our fully assembled quadcopter. We tested the quadcopter response without propellers installed, tethered with propellers , in air with propellers and submerged underwater (testing videos attached in **Appendix**) The current prototype is capable of lifting off from ground, hovering in the air, and adjust states(x,y,z,roll,pitch and yaw) in the air. In addition, the current prototype is fully waterproofed, meaning we are able to control the quadcopter when it's immersed in water and the system is still fully function after we take it out from water.

Limitations and Concerns

One of the limitation of the system is that our WiFi module is only able to guarantee data transmission when the quadcopter is around 20-25cm underwater with antenna face upward due to the reason that 2.4GHz microwave tends to be absorbed remarkably in water. In addition, battery waterproofing also raises a safety concern. Currently, we taped around all the edges of the battery and apply hot glue to connections and all the places that are vulnerable to potential leakage. However, battery may be damaged if the tapes or hot glue wear off. Furthermore, we need determine the new motor speed threshold for the system underwater. The reason is that full speed is not needed for the quadcopter to lift off and hover in air, but the speed threshold has to be readjusted due to higher resistance in water to achieve smooth movement of the quadcopter underwater.

Another concern of the quadcopter during the in air testing is the unwanted yaw rate that caused the quadcopter to spin during hovering. This issue is mainly caused by the accuracy of the MPU6050, whose readings still have some noise after calibration. This inconsistency is further exacerbated by the small size of the quadcopter. Because this quadcopter is very light, a small change on PWM signal of one motor can cause great disturbance on the quadcopter. This problem can be solved by very precise tuning of the PID coefficients on the yaw rate controller, but we didn't have enough time left to finish the PID tuning. Also, as mentioned above in the control section, currently the controller is only able to send one set of PID constants at a time. We'd like to have more robust control on yaw rate, pitch and roll separately, so ideally there should be three sets of PID constants in one packet sent through the controller.

Finally, the PID constants in air and in water are different because of different environmental resistance. Right now we are able to manually change the PID coefficients through the serial monitor demands depending on the environment that the quadcopter is in, but the transition is mostly done by visual observation, which is not ideal. In order to automatically switch from "air mode" to "water mode" and change the PID constants when the environment situation changes, we need another input from an additional sensor, such as a water sensor.

Future Improvements to be implemented:

- Add a mini camera so that we can achieve visual maneuvering of the drone
    - https://www.banggood.com/FPV-5_8G-200mW-32CH-Wireless-Audio-Video-AV-Transmitter-Module-FX758-2-p-980317.html?p=4S26211552037201504F
    - https://www.banggood.com/600TVL-8_0MP-14-2_8mm-CMOS-FPV-170-Degree-Wide-Anlge-Lens-Camera-PALNTSC-p-984345.html?p=4S26211552037201504F
- Use more robust controller for more precise control
- Fix Yaw Drift or use more advanced IMU
- Sensor to determine whether in air or in water

Conclusion

Again, our goal was to break the boundary of the operational spaces of the drone. Through the preparations of last quarter and the implementations of the spring quarter. Our amphibian quadcopter is be able to launch and hover shortly in air and conduct various modes of underwater controls through wireless signals. The PID controller of the quadcopter functions correctly and optimize the controls of quadcopter in air and underwater. All electronic components are completely waterproof-coated and free of any types of environments. During the classes of EE 183DA and EE 183DB, we learned the fundamentals of robotics and robotic controls. Our team has studied a vast range of topics, such as DMP, PWM with motor control, system and control design, common electronic hardwares and wireless communications during the preparation process. We independently designed and achieved essential realizations of the amphibian drone in the span of last two quarter. We wish our works can inspire future engineers to dig deeper into this topic and hopefully we have paved the beginning of the road for people who are willing to carry this project further. We realized that there are both opportunities and limitations residing in the technologies and techniques our team used; yet, it is our motivation to push the boundaries of our knowledge and to experiment with existing technologies we are bestowed. We were extremely fortunate to have the opportunity to make our idea of an wireless amphibian drone come into reality and we are in appreciation of the help and feedbacks from our classmates and instructors. Further demonstration of the results of our project and source codes can be found in the **Appendix**, all under MIT license.

Reference

- Franklin, G. F., Powell, J. D., Emami-Naeini, A., & Sanjay, H. S. (2015). Feedback control of dynamic systems.
- PWM Wikipedia page: https://en.wikipedia.org/wiki/Pulse-width_modulation
- Tech Talk : Quadcopter Basics. (n.d.). Retrieved June 16, 2017, from http://www.xybernetics.com/techtalk/QuadcopterBasics/QuadcopterBasics.html
- Micro 105 FPV Quadcopter - 8.5mm Motors, Micro Scisky: https://www.thingiverse.com/thing:1221911
- Quadcopter motor placement http://quadcoptergarage.com/upgrading-my-quadcopter-from-kk2-to-apm-2-5/
- WiFi Module Datasheet https://www.sparkfun.com/datasheets/Components/SMD/nRF24L01Pluss_Preliminary_Product_Specification_v1_0.pdf
- IMU Datasheet https://www.invensense.com/products/motion-tracking/6-axis/mpu-6050/
- ATmega 328P Datasheet http://www.atmel.com/Images/Atmel-42735-8-bit-AVR-Microcontroller-ATmega328-328P_Datasheet.pdf
- Voltage Regulator Datasheet https://cdn-shop.adafruit.com/product-files/3081/mic5219.pdf
- Controller Code Repository https://github.com/cullenmq/Controller

- DMP Library
  https://github.com/jrowberg/i2cdevlib

Appendix

Quadcopter control code

https://github.com/duanzr1996/Quadcopter

Video of quadcopter vertical lift-off

https://youtu.be/mVXtSzUjBPE

Video of quadcopter hovering

https://youtu.be/wt5oHvoHgVM

Video of underwater testing

https://youtu.be/BKowKTW0-io

Parts List

https://docs.google.com/spreadsheets/d/1l4aeeZpLU2HqdHF7owFncWuhiHhhJrN1-ExrxptnZMY/edit#gid=0