

# Crazyflie Swarm Research

Team Number: 1732

Advisers: Nicola Elia, Phillip Jones

## Team Members/Roles:

Nick Robbins - Project Lead

Cole Beaulieu - Communications Lead

Jacob Frazier-Florres - Web Design Lead

Grant Manley - Co-Webmaster

Ben Nelson - Concept Leader

Tianxiang Shen- Co-Idea Leader

Chengrui Yang- Co-Idea Leader

# Contents

## 1 Introduction

1.1 Project statement

1.2 Purpose

1.3 Goals

## 2 Deliverables

## 3 Design

3.1 System specifications

*3.1.1 Non-functional*

*3.1.2 Functional*

3.2 Proposed Design/Method

3.3 Design Analysis

## 4 Testing/Development

4.1 Hardware/software

4.2 Process

## 5 Results

## 6 Conclusions

## 7 References

## 8 Appendices

# 1 Introduction

## 1.1 PROJECT STATEMENT

The Quadcopter swarm project revolves around creating a swarm of Crazyflie quadcopters. As it stands currently, the Crazyflie open source platform enables the user to fly one or two crazyflies with one included radio. In order to fly more, we need to change how the radio and ground station process these crazyflies. Once we get more than two flying we are going to also implement crazyflie to crazyflie communication. We also have the benefit of a camera system to tell us exactly where these crazyflies are located.

## 1.2 PURPOSE

The implementations for this project are extremely open. With a swarm of quadcopters being controlled as a group or individually the user could achieve things like carrying a heavy or strangely shaped load, choreographing a multi camera scene, as well as recreational uses like programming them to perform aerial acrobatics. ( See CrazyFlie wiki for details and history)

## 1.3 GOALS

We have multiple goals that will build upon each other:

1. Implement a wifi adapter instead of using the Crazyflie Radio
2. Create a ground station that uses UDP communication
3. Implement the BigQuad expansion deck to expand crazyflie size.
4. Enable Crazyflie to Crazyflie communication.

# 2 Deliverables

Hardware:

- A new fully customized BigQuad that flies using the crazyflie base.
- Integrate ESP WiFi chip into ground station
- Create expansion deck to easily connect Wifi module to the crazyflie deck

Software:

- New firmware for the ESP WiFi chips that is capable of point to point, broadcast, and mesh communication (receiving and transmitting).
- Firmware adaptation for the crazyflies that allow them to receive communication from the ESP Wi-Fi chip using uart pins pre-built into the crazyflie.
- New ground station code that enables communication over the new Wi-Fi device. ( See Bitcraze Documentation for more information)

## 3 Design

The design of the project will revolve around the ability to produce a large quad-copter while at the same time creating a swarm controlling system that can fly multiple quads at the same time. The main focus areas will be around the construction of the quad, radio communications, and electrical design systems for the power systems.(See Crazyflie Wiki to see what has been done in the past)

In order to achieve the project goals we have two options: make alterations to the current communication stack or scrap the current system and implement something new.

Current System:

Benefits:

- Utilizes previous work done on the project, which could potentially save time
- Doesn't require the purchase of any new communication links.

Negatives:

- System was not designed for flying multiple quads at once and therefore is not easily translatable to our project.
- Lack of expandability for Quad to Quad communication

To alter the current system we would either need to be changing frequencies on the fly or implementing another form of multiple access (TDMA, CDMA). Changing frequencies on the current hardware proved to consume too much time to reliably fly the quads and the other forms of multiple access were either too complex or impossible to implement on the current radio. For these reasons we elected to scrap the current communication system and implement one chosen to specifically satisfy our needs.

After researching multiple options, we decided to implement the system using ESP8266 Wifi modules because our lab has had success using them in the past, they are simple to implement, and they work in a similar fashion to the current crazyflie radios, easing the conversion process.

### 3.1 SYSTEM SPECIFICATIONS

The specifications that we are given to adhere to in this project are:

- Create a large scale quadcopter from scratch using non-custom parts and hardware
- Redesign the radio communication systems to be able to talk to multiple quadcopters with a given radio dongle
- Implement code into firmware so that quad to quad communication can occur if necessary( See Bitcraze Documentation for more information)
- Use the camera systems provided to help log data with the tracking systems for giving 3D spacing requirements

#### 3.1.1 Non-functional

- Small diameter quads to fit up to 10 at once in the lab environment

- Nearly zero complete failure (crashing) rate
- Low command failure rate (failing to follow a command without crashing)
- High data transfer rate for quad to quad and ground station to quad communications
- Easy to use user interface

### 3.1.2 Functional

- Give precise commands to each quad individually
- Broadcast messages to all quads
- Quad to quad communication

## 3.2 PROPOSED DESIGN/METHOD

Our proposed design method is to initially use the provided the hardware and software that is given to us and initially figure out how it works. Then we will try to modify the code either by introducing new hardware and software with new units until we get the ability to fly multiple large quads with a single dongle at the same time. The Software team of the project is currently looking in streamlining the communication software to improve the ability of multiple communication points. While the Hardware team is building a new quad that will have faster and better hardware to enable the software team to better improve their systems.

## 3.3 DESIGN ANALYSIS

Our initial approach was to modify the current ground station software to improve the efficiency of the CrazyRadio Dongle. When we started, the radio was communicating to different quadcopters by switching to a unique channel for each quadcopter. Channel switching is a costly operation, so we thought that by keeping all of the quadcopters on the same channel and using separate addresses, we would be able to support more quadcopters. We also wanted to remove the acknowledge packet used in the current protocol to improve performance.

However, as we tried to implement these changes, we discovered several things:

1. The library we were using to interface with the CrazyRadio in C++ was not fully implemented, poorly documented, and not designed to handle modification well. This made all changes to the ground station software take significantly longer than expected.
2. The CrazyRadio Dongle is difficult to interface with.
3. The acknowledge packet is integral to the operation of a lower-level protocol used by the system, and its removal would be non-trivial.

These discoveries were not all bad news, however. This work gave us a much more thorough understanding of the system as a whole, including the fact that it would likely be possible to totally bypass the current radio setup. We decided to use small, inexpensive ESP8266 WiFi module, which we could connect to the ground station and to the STM32F405 microcontrollers on the quadcopters.

These WiFi modules allowed us to build a new communication system between the Ground Station and the Crazyflies which is more extensible for future projects. The firmware on the WiFi modules is simple to write and the process of flashing new firmware is relatively painless. This will allow future teams to experiment with different communication setups, including broadcast messaging, a mesh-network, and quad-to-quad communication. The new firmware that was written for the Crazyfly to support the new WiFi module also allows us to switch to a different model of WiFi chip easily.

## 4 Testing/Development

### 4.1 HARDWARE/SOFTWARE

Indicate any hardware and/or software used in the testing phase. Provide brief, simple introductions for each to explain the usefulness of each.

#### **The hardware that we are using for testing this project include:**

Arduino Uno - Enables us to power the wifi module as well as pass data through a serial port

Breadboard - Enables us to set up the circuit needed to flash firmware onto the wifi module

Function Generator - Used to test the motors and esc that will be used on the Big Quad

FTDI Programmer - Used to connect the computer's usb port to the wifi module to view packets being sent to the module

Multimeter - Used for various parameter measurements

Tachometer - Used to measure rpm of motor

Oscilloscope - Used to analyzed duty cycle of crazyflie thrust commands

#### **The software that will be used in the development of this project include:**

ESP8266 Mesh Library - A previous mesh code was used to test if our modules have the capability for connecting to multiple other modules ( See ESP8266 Library for more detail)

Current C Client Code - This was used as the basis for what we build everything off of.

### 4.2 PROCESS

#### **Testing Broadcast and Individual Messaging:**

Figure 1 displays the setup our team used to test the ESP Modules' messaging capabilities. Our team sent a message from one ESP and monitored the reaction of two other ESPs' using a serial reading program on a PC. During the first test our team used code designed to send data to a single, specific ESP on the network and verified that only that ESP received the data. During the second test, our team used firmware designed to broadcast data to the entire network and confirmed that both ESPs received the broadcasted message. While these test were simple they were import in the early stages of the project in order to verify that the new communication hardware was capable of achieving the basic functionality needed for our requirements.

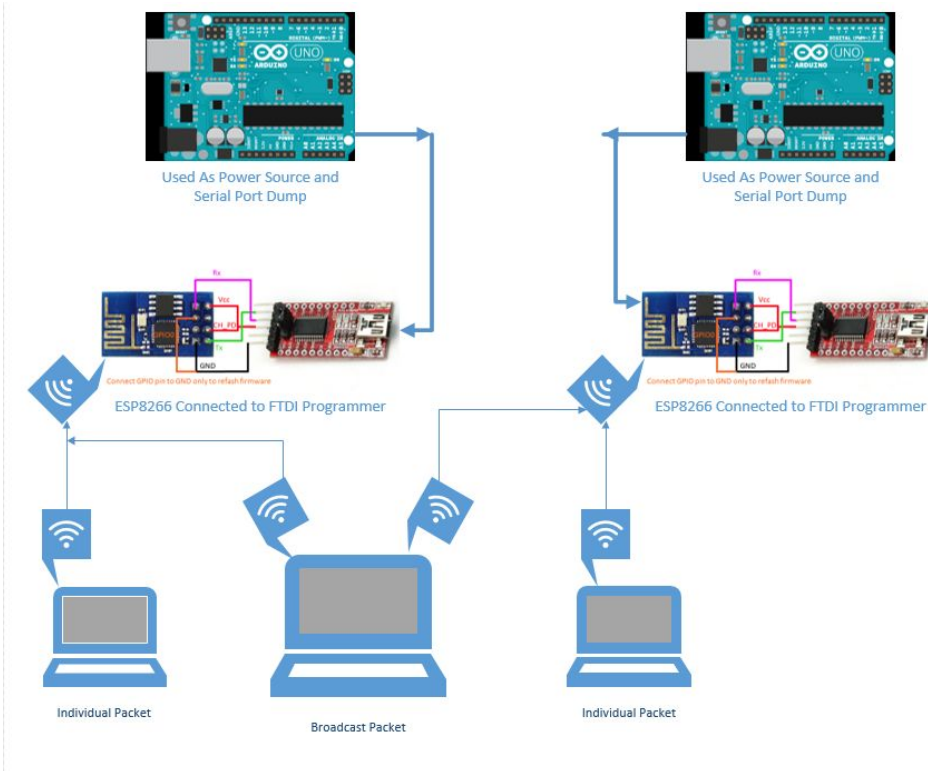


Figure 1

### Testing Multi Module Mesh Networking:

Figure 2 shows how our team tested the mesh network capabilities of the ESP modules. As shown below, our team configured the setup to send a packet from the ground station computer to one of the modules. This module then relayed the same packet on to the next module. This module then responds with an acknowledgement that it received the packet and that acknowledgement relayed back to the ground station computer. This test showed a basic proof that the crazyflie's communication range may be increased via message forwarding.

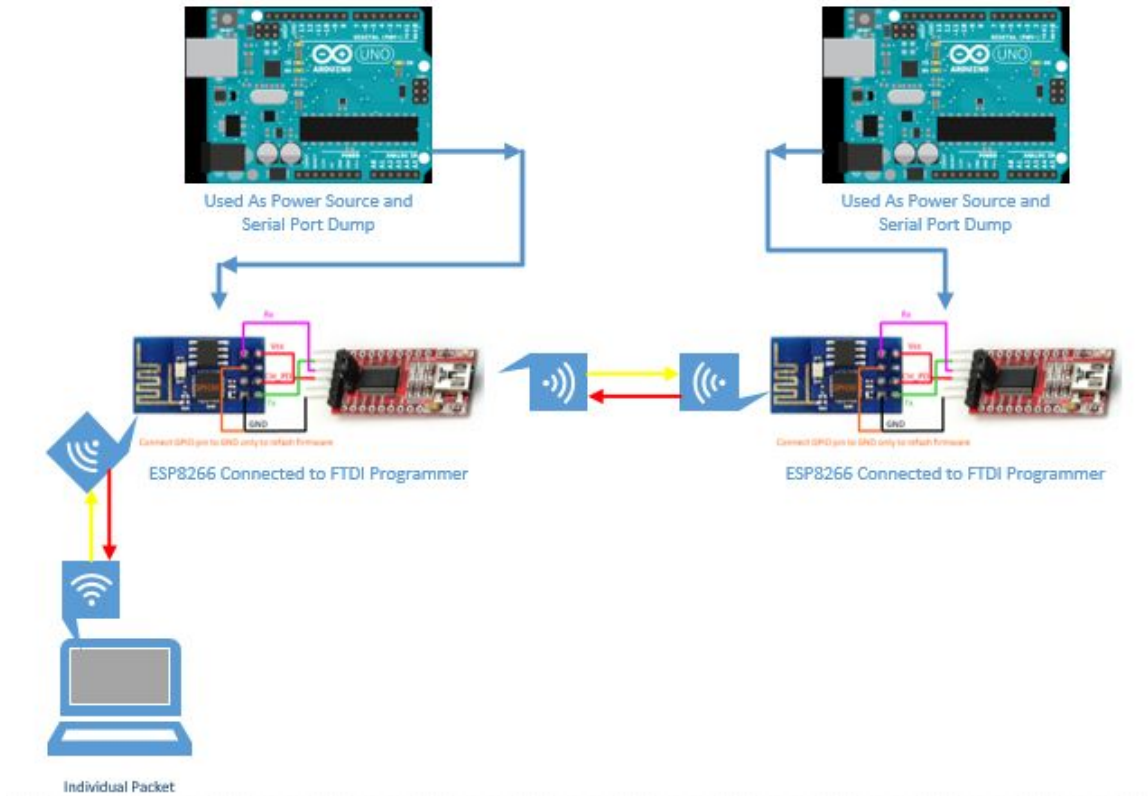


Figure 2

### Testing Crazyflie Firmware

In order to test the crazyflie firmware, our team measured the time it took to receive a response after a ping data packet was fed into the uart RX port. The average response time over 1000 packets was ~42 ms. Our team also tested the crazyflie’s responsiveness to different kinds of packets. The most important for our requirements are known as setpoint packets and instruct the crazyflie on it’s flight controls such as thrust, pitch, roll, and yaw. The crazyflie responded correctly to setpoint packets, as well as all other packet types pertinent to our system.

We also tested the latency for just the ESP to ESP communication. For these tests we eliminated the latency due to the Crazyflie and the Serial connection between all the various components. We found that, on average, there is a 2.52ms round trip latency between two ESP modules.



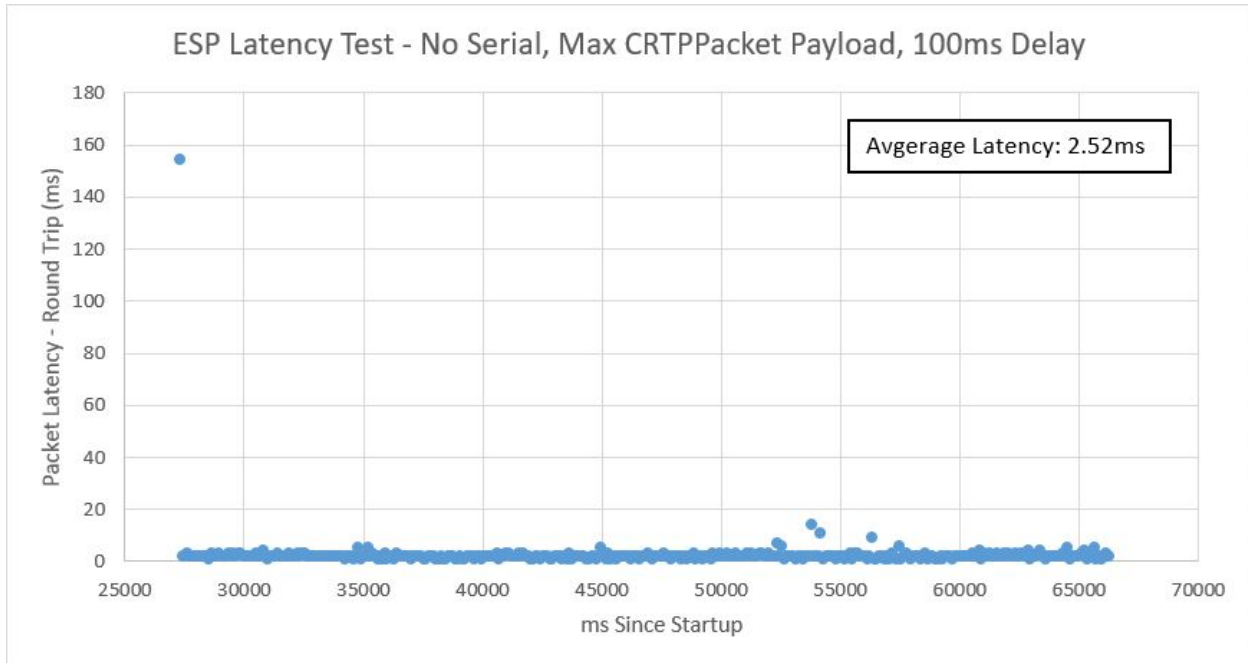


Figure 4

### Testing Autonomous Flight with ESP Module

The final testing conducting on the new communication stack involved linking the Crazyflie firmware with the ESP based ground station to achieve autonomous flight. This test uses a VRPN camera system to track a Crazyflie's position and orientation, relays this data to the ground station, the ground station then makes calculations based on that data, and sends setpoints to the Crazyflie in order to control its flight.

### Testing Big Quad ESC and Motors:

To test if the ESC and Motors we are going to implement on our big quad were functioning, we connected them to a batter to serve as the power source as well as a function generator to mimic the function giving off by the controller we are using. We used the below settings for our function generator.

Amp: 3.3V

Freq: 400 Hz

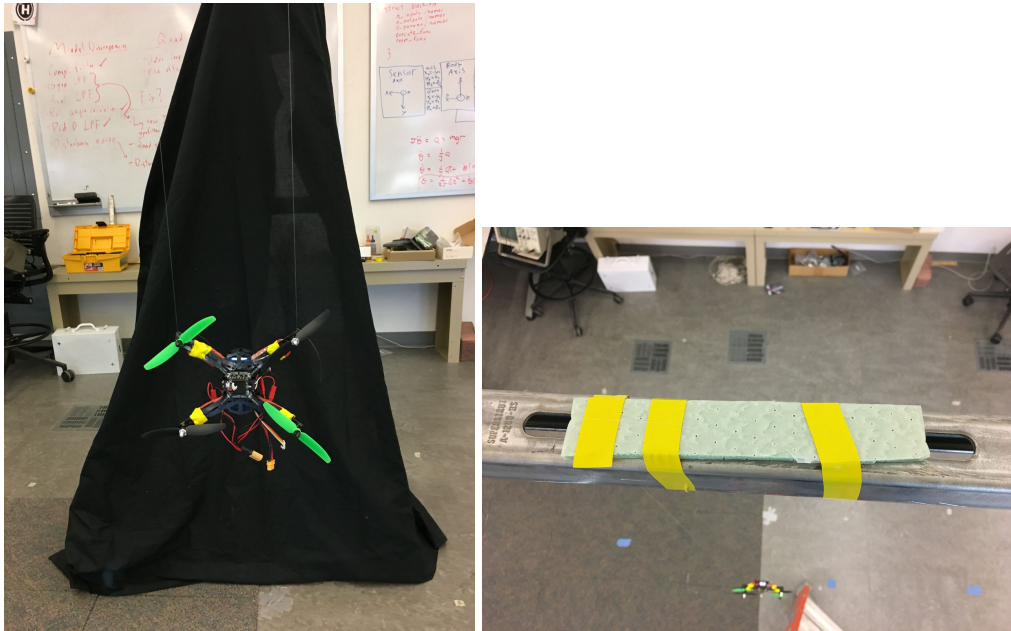
Duty Cycle: 40%

Offset: 1.6V

We were able to see each of the 4 motors spin along with varying speed by adjusting the duty cycle.

## Big Quad Parameter Gathering:

In order to simulate flight using a grad student in our group's matlab program, we needed specific parameters about the quad so that it could emulate flight compared to the current Crazyflies. We gathered all the necessary parameters using resources like our grad students, a prior thesis, and our own research. For instance, in the figure below there is a setup for finding the moment of inertia of the quad by counting oscillations across one axis.



Figures 4 & 5

Here are the requested parameters and the results:

(1) mass and center of mass

**296g with new battery**

**COM – 2 inches from bottom plates length edge, side with the standoff screwed in**

(2) moments of inertia for each axis  $J_{xx}$   $J_{yy}$   $J_{zz}$

**Pitch .000699**

**Roll .000695**

**Yaw .001259**

(3) motor+propeller moment of inertia ( this specifically has a different method than the whole body MOI)

**$1.98278 \cdot 10^{-5}$**

(4) propeller thrust constant

**Kt:  $2.6282e-07$**

(5) propeller drag constant

**Kd:  $7.8488e-05$**

(6) positions of propeller hubs w.r.t. center of mass (in a local frame of reference where +x is "forward", +y is "right" and +z is "down").

	X (inches)	Y (inches)	Z (inches)	Calculated $\sqrt{x^2+y^2}$ perp length	Actual $\sqrt{x^2+y^2}$ diagonal length from COM
M1	3	3.4375	-2.5625	4.4375	4.5625
M2	-3	3.4375	-2.5625	4.4375	4.5625
M3	-3	-3.4375	-2.5625	4.4375	4.5625
M4	3	-3.4375	-2.5625	4.4375	4.5625

Table 1

(7) motor resistance

**.335 ohms**

(8) motor Kv

**3100 kv or**

**324.63 in rad/s**

(9) motor friction current (either an overall average or function of speed)

**.4 A**

(10) ESC turn on and maximum duty cycle input

**ESC turn on:25 %**

**Max duty cycle: 68%**

### **Manual Flight:**

To check how stable our quadcopter would fly, we began by checking how easy it was to fly manually. We designated Jake to become an expert in flight with the little Crazyflies so he could pilot our Big-Quad. Using the default Crazyflie PC Client, we were able to fly successfully. However, we noticed our quad was not extremely stable. To combat this issue, we began tuning our PID for both rate and attitude.

### **Tuning Big Quad PID Rates and Angles:**

After we checked manual flight, we moved over to using the camera system in the lab to fly autonomously. On our first attempt of autonomous flight we ran into problems. Our quadcopter, while

somewhat stable under manual flight, became extremely unstable for autonomous flight. This led us to believe that we need to tune the inner loops of the PID.

We tried several different set ups to isolate the three different axis so that we could tune the roll, pitch and yaw individually. Pictured below is one of the set-ups we used to tune the pitch and roll rates. We were able to find values so that we could fly the quadcopter with just thrust input without the need of pitch and roll.



Figure 6

## 5 Results

### Hardware:



Figure 7

Implementing the BigQuad expansion board: We received our expansion board and implemented it on our newly built quadcopter. Using the PID values below, we were able to easily fly the quadcopter by hand. The quad is responsive and stable. The only problem we noticed with manual flight involves when

the battery runs low. There is no notification given when the battery runs low. If the battery is too low, the quadcopter has the possibility of breaking the connection to the ground station and flying out of control.

PID values used for manual flight:

<b>PID values table</b>	Rate	Angle/Attitude
Kp Roll	110.0	5.0
Ki Roll	0	3.0
Kd Roll	0	0
Kp Pitch	110.0	5.0
Ki Pitch	0	4.0
Kd Pitch	0	0
Kp Yaw	0	10.0
Ki Yaw	25.0	0
Kd Yaw	50.0	2.0

Table 2

### **Autonomous Flight:**

Based on the values gathered above and our successful manual flight tests we decided to attempt autonomous flight using the groundstation. We entered in the values we gathered above into the ground station and attempted flight. However, we ran into a variety of issues. The most pressing issue is how this camera system groundstation treated the flight in our quadcopter. The thrust commands that are sent to the quadcopter are extremely violent on takeoff, and do not mimic manual flight. This extreme thrust input would immediately cause the quad to become unstable. In order to fly a quad of this size, a new flight ground station would need to be created in order to mimic the way a person would fly the quadcopter. Another possible solution for this problem would be to use manual control on takeoff, and then switch over to autonomous flight after a certain altitude is reached.

### **Software:**

#### **Radio communication:**

The original communication stack for the Crazyflies utilized radio communication and frequency division multiplexing. This was problematic for flying more than 4 Crazyflies at a time, because the latency involved in changing the frequency became too high for autonomous flight. After researching other forms of multiplexing, it was concluded that the out-of-the-box crazyflie radio was not designed for the requirements our team was trying to impose on it, so it was decided that the system should be replaced in favor of an ESP WiFi module.



Figure 8

### **Wifi module:**

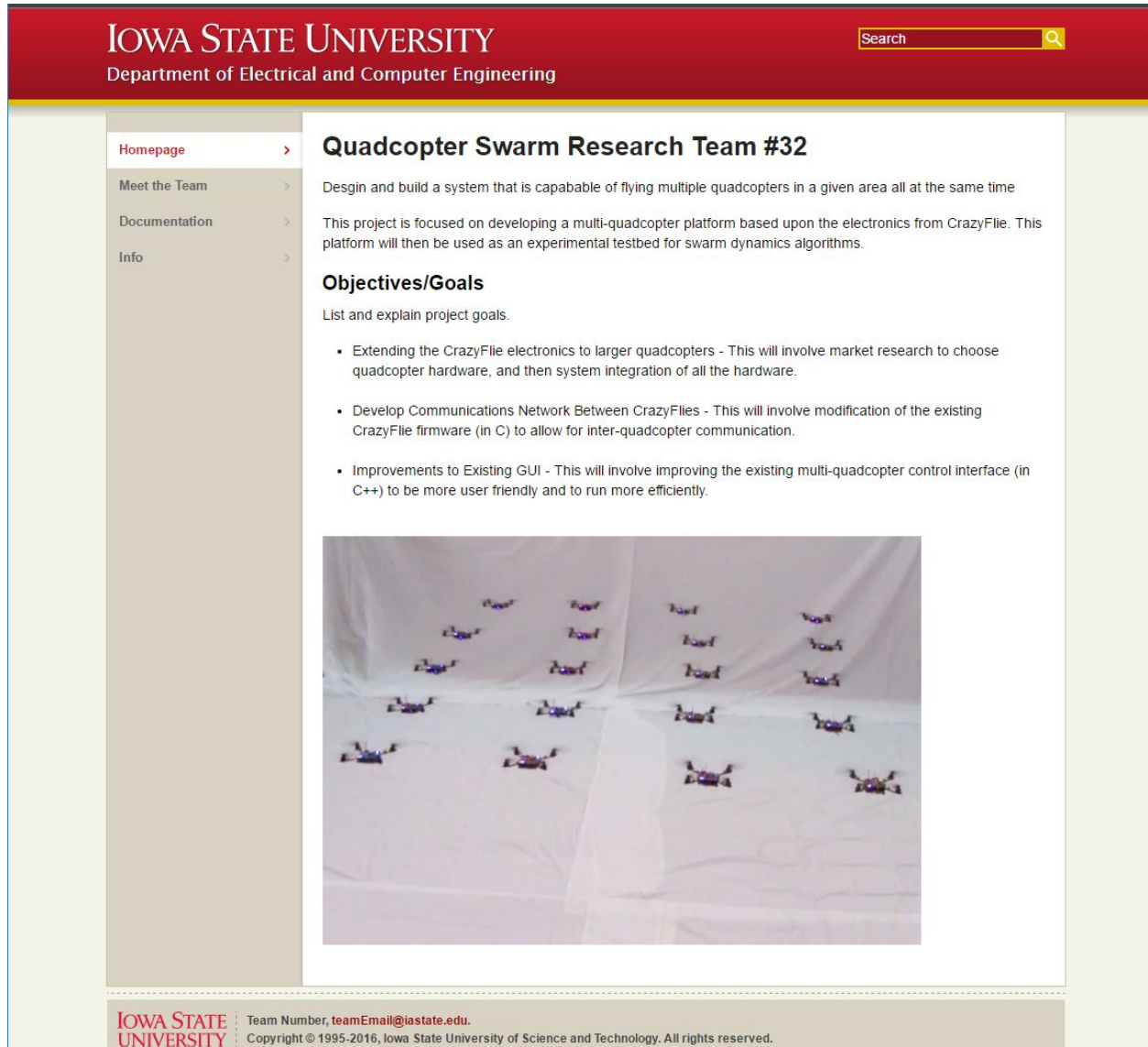
The WiFi modules have been successfully implemented into the Crazyflie communication stack on both ends (ground station and firmware). The firmware our team implemented on the ESP modules is capable of receiving and transmitting messages to one specific module on the network or to all modules connected to the broadcaster. They are also capable of communicating via UART with both the ground station and the crazyflies. Furthermore, the ESP modules can support up to 8 connections, making them ideal for flying a swarm of crazyflies simultaneously. While it has not been live tested, simplified testing has also shown a potential for mesh network capabilities, that would allow our lab to extend the communication range of the crazyflies by relying on messages through in-range crazyflies.

### **Crazyflie Firmware:**

The Crazyflie firmware has been successfully adapted to receive and transmit messages via the ESP WiFi module. The Crazyflie firmware is based on an open source project titled FreeRTOS (<http://www.freertos.org/>) and to implement WiFi communication under this system a new task (similar to an infinitely running process) was created to constantly poll data over the UART port. When data is received it is verified for correctness and then passed along the same channels used by radio communication packets. This allowed for minimal changes to the existing firmware while still accomplishing the objective. For sending data over WiFi communication, Crazyflie's firmware already had a built-in system for switching between radio and Bluetooth transmissions, so once a WiFi communication driver was implemented it was simple to switch transmission methods from radio to WiFi. All of these changes are implemented alongside all other forms of communication and switching between radio and WiFi communication can be accomplished by simply changing a flag in the config.mk file. Overall, this new functionality is completely modularized and will not affect any other changes made to the Crazyflie firmware.

## Website:

The website is up and maintained all of our weekly reports and documentation that relates to the project. All the information about the project and team members are welcome to check out though this website. The below figure 9 shows the homepage of our website.



The screenshot shows the homepage of the Quadcopter Swarm Research Team #32 website. The header features the Iowa State University logo and the Department of Electrical and Computer Engineering. A search bar is located in the top right corner. The main content area is divided into a left sidebar with navigation links (Homepage, Meet the Team, Documentation, Info) and a main content area. The main content area includes a title "Quadcopter Swarm Research Team #32", a description of the project, a section for "Objectives/Goals" with a list of three bullet points, and a photograph of several quadcopters flying in a formation. The footer contains the Iowa State University logo, team contact information, and a copyright notice.

**IOWA STATE UNIVERSITY**  
Department of Electrical and Computer Engineering

Search

**Homepage** >  
Meet the Team >  
Documentation >  
Info >

### Quadcopter Swarm Research Team #32


Design and build a system that is capable of flying multiple quadcopters in a given area all at the same time

This project is focused on developing a multi-quadcopter platform based upon the electronics from CrazyFlie. This platform will then be used as an experimental testbed for swarm dynamics algorithms.

#### Objectives/Goals

List and explain project goals.

- Extending the CrazyFlie electronics to larger quadcopters - This will involve market research to choose quadcopter hardware, and then system integration of all the hardware.
- Develop Communications Network Between CrazyFlies - This will involve modification of the existing CrazyFlie firmware (in C) to allow for inter-quadcopter communication.
- Improvements to Existing GUI - This will involve improving the existing multi-quadcopter control interface (in C++) to be more user friendly and to run more efficiently.



**IOWA STATE UNIVERSITY** Team Number, [teamEmail@iastate.edu](mailto:teamEmail@iastate.edu).  
Copyright © 1995-2016, Iowa State University of Science and Technology. All rights reserved.

<http://may1732.sd.ece.iastate.edu/>

Figure 9

## 6 Conclusions

So far we have tested and proven the feasibility of almost all aspects of our proposed design including: utilizing the ESP8266 Wifi module to build a local network, evaluating the crazyflie firmware ability to adapt to the new communication link, and building an enlarged quad prototype to show the new hardware working in unison. These tests have proven that we will be able to use our proposed system to build a communication network capable of flying upwards of 8 quads at once and use them to perform uniformed tasks. It has also shown that quad to quad communication is a feasible goal for future iterations of the project. Furthermore, our proposed solution of building a new communication stack is the best solution for the following reasons: the current communication stack is incapable of reliably achieving our goals, the Wifi modules are specifically designed to build local networks of this type therefore making the implementation of such easier on our software team, and the Wifi modules are similar enough to our current links to ease the adaptation process in our firmware and ground station. With BigQuad expansion deck, we are able to expand the size of crazyflie by implementing it on our self-designed Big quadcopter. Through plenty of tests, we ensured the reliability of connections about our BigQuad connecting with crazyflie base. All the parameters we obtained in the lab including thrust constant, moment of inertia and etc made manual flight successfully but poor stabilization is still a problem. Tuning the PID value is a major part for the hardware team. The stabilization is deeply improved by setting new PID for manual flight. The current existing problem is low quality of stabilization when operating autonomous flight using the groundstation no matter how to further adjust the PID values and other parameters..

## 7 References

ESP8266 Library: <https://github.com/esp8266/Arduino/blob/master/doc/esp8266wifi/readme.md>

Bitcraze Documentation: <https://www.bitcraze.io/>

Crazyflie Wiki:

[https://wikis.ece.iastate.edu/distributed-autonomous-and-networked-control-lab/index.php/Crazyflie Swarm](https://wikis.ece.iastate.edu/distributed-autonomous-and-networked-control-lab/index.php/Crazyflie_Swarm)



## 8.1 Appendix I: Operation Manual

### BigQuad Manual Flight

- 1) Begin by downloading and installing the bitcraze virtual machine located [here](#). This virtual machine comes preloaded with the crazyflie client as well as the firmware.
- 2) Once in the virtual machine, click the UPDATE PROJECTS icon on the desktop. This will ensure you have the newest firmware.
- 3) Next, you will need to enable the BigQuad Expansion Deck in the firmware. Directions for this step can be found [here](#).
- 4) Once you have updated the firmware to include this expansion deck, you can flash this firmware to the Crazyflie. To do this, make sure the Crazyflie is powered off and then hold the power button down for a few seconds. When you see alternating blue lights, the Crazyflie is in bootloader mode.
- 5) You can now go into the Crazyflie Client GUI and follow the directions [here](#) to flash the firmware to the Crazyflie
- 6) Once the firmware has been flashed, restart the Crazyflie quadcopter. Then, connect to it through the GUI once again. Before flying, you should verify that you have the correct PID constants.
- 7) To do this, go into View→Parameters and scroll down to until you see pid\_attitude and pid\_rate. Change these values to match what is in the RESULTS section of this document.
- 8) Lastly, plug in a PS3 controller to fly the quadcopter. (Note: it may take a few tries to fully understand how to fly the Crazyflie).

## 8.2 Appendix II: Alternate design versions

During the design process we had two different models of designs for the quadcopter, we had a small one where we tested the mesh network on for the communication stack, and a larger one where we were testing basic flight controls to get them tuned and autonomous. Initially for the design of the project was for us to take a couple of the small quadcopters and learn how to fly them in sync. We discovered software/hardware limitations which restricted us from doing that, this lead to a change in the client's specs to a bigger quad with more capabilities. With the bigger quad in play we had to redo a few things ( tuning and testing evaluations) which didn't produce the expected results. This project was not a "Cut and Dry" project it had many different design items that often changed every 2 weeks , to see if there were better options available for each sub design

At the beginning of the project we were given the base code for the communication stack and the base hardware for the quadcopter. We were not given an initial choice in what we would be using. However we were given the choice of the bigger quadcopter parts, however the controls had to remain the same. This eliminated the option for commercially available quadcopters that had a better controls systems and that

were open source. The client has been informed that if they would like to continue the project it might be a good idea to allow standard industry software and premade hardware to be allowed.

### 8.3 Appendix III: Other Considerations

Over the last 9 months, the team has encountered many different challenges and physical problems. These range from multiple firmware being shared between quadcopters among multiple senior design teams, lack of documentation for the camera systems and ground station for controlling the quadcopters, and physical performance boundaries of the parts. Most recently there has been products released on the market that are better basis for the project next year (Parrot Quadcopters), there has been discussion with the advisors on whether we should continue the project with the current system that is a few years old to a new system that has more advanced capabilities with better documentation.