# Knight's Intelligent Reconnaissance Copter - KIRC

Nathaniel Cain, James Donegan, James Gregory, and Wade Henderson

Dept. of Electrical Engineering and Computer Science, University of Central Florida, Orlando, Florida, 32816-2450

***Abstract* — A senior electrical and computer engineering project, the Knight's Intelligence Reconnaissance Copter – KIRC is quad rotor unmanned aerial vehicle with incorporated features such as delay tolerant networking, custom flight control algorithms, and image mapping of a predefined area. This paper presents the methodology to realize appropriate hardware and software components which together satisfy the requirements for the above mentioned system.**

***Index Terms* — Quadcopter, Delay Tolerant Networking, Flight Control Systems, Aerial Imaging, autonomous.**

**Figure 1: 3D Model of Quadcopter**

## I. Introduction

The Knights Intelligence Reconnaissance Copter project, unofficially sponsored by the National Aeronautics and Space Administration (NASA), started as an idea to show the potential usefulness of an emerging networking protocol called DTN. Delay Tolerant Networking (DTN) is a virtual networking protocol that excels in areas where communications networks are prone to delays and disruptions. DTN is of particular interest to NASA for the potential for deep space missions where there are long distances and intermittent links between spacecraft. The project also aims to advance earth science missions by using unmanned aerial vehicles to image an area from the sky. The project has the potential to be used for imagery in upcoming NASA missions such as the Pavilion Lake Research Project in Canada in the summer of 2014.

One of the important details in the project is that the team decided to create the quadcopter from scratch. The reason behind making a custom built quadcopter versus buying a commercial off the shelf (COTS) quadcopter is that the custom control system can make a smooth transition into autonomous guidance easier. With COTS items, many vendors will not make the software open source to the public. This complicates the design of the guidance control algorithm significantly, and also makes the software less predictable. Another reason is that quality COTS quadcopters can tend to be quite expensive; since the unofficial sponsorship from NASA provided a budget of $1000 for certain parts, price became an important factor throughout the project design. With all this in mind, it was decided that it was in the team's best interest to design a custom quadcopter for this project. It is to be understood that autonomous guidance of the quadcopter is a reach goal of this project, the main goal of the project is to successfully build a custom quadcopter with a custom stability control system, uses DTN to communicate with a ground station laptop, and can image an area that a user specifies.

While the Pavilion Lake Research Project is the initial application of this project for NASA, it will also be planned to be used in future projects as well. For this reason, the project system should be designed to be reconfigurable based on open ended mission needs. A long term goal of the project is to provide a modular payload that can be a platform for the core services of this project, but be adaptable to other UAVs.

## II. System Components

Each component for this system was chosen based on several factors which often yielded tradeoff characteristics. This section provides a description of the components used as well as what factors were taken into consideration when choosing said components.

### *A. Microcontroller*

The first and most significant component chosen is the microcontroller because it has the most impact on the software portion of the project. The heart of the stability control system is the TM4C123GXL Tiva C ™ microcontroller from Texas Instruments. A 32 bit, 80MHz, ARM Cortex M4 Processor along with 32KB available RAM and 256KB flash ROM proved ideal to implement the necessary real time operating system

(RTOS) needed for the quadcopter. One very attractive feature of this microcontroller is the RTOS available from Texas Instruments. The Tiva ™ C has a Launchpad available for prototyping as well as a standalone microcontroller chip for integration into the printed circuit board (PCB). The microcontroller also has support for multiple UART, PWM, I2C, A/D, and GPIO ports in hardware, sufficient to support any type or amount of data from the sensors.

*B. Inertial Measurement Unit*

The inertial measurement unit (IMU) is an important component containing almost all of the on board sensors. Four sensors make up the IMU for this project: the ADXL345 accelerometer from Analog Devices, the HMC5883L magnetometer from Honeywell, L3G4200D 3-axis gyroscope from ST, and the BMP085 barometric pressure and temperature sensor from Bosch. All the sensors have support for I2C master-slave communication to communicate with the microcontroller. The accelerometer and gyroscope are used to recognize orientation changes of the quadcopter for stability control, the barometric pressure and temperature sensor is used to determine the altitude, and the magnetometer is used to determine exact relative rotation to a remote location in terms of degrees for autonomous navigation. These sensors are all available for individual purchase from their respective companies and are purchased to integrate onto the PCB.

*C. GPS Sensor*

The GPS sensor stick GP-635T produced by ADH Technology Co. Ltd. was chosen because it proved to have the best performance for the price. The GP-635T has the ability to use to 50 satellites at one time, a low sensitivity of -161 dBm used for precise tracking and subsequent navigation, and a cost effective price at $39.99 each, half the price of other competitors. The cold start time for the GPS is 27 seconds so the quadcopter will have to stay on the ground at least that long before the GPS is ready for flight. The GPS transmits data to the microcontroller using UART serial communication at 115200 baud, 8 data bits, 1 stop bit, and no parity.

*D. Navigation Computer*

NASA required the team to use a DTN, which only runs on Linux. The team elected to use the Raspberry Pi ™ Model B, an embedded Linux platform, to test the DTN application as well as to handle the image retrieval and telemetry relay. Conveniently the Raspberry Pi ™ also has an available 5 megapixel camera attachment made specifically for the Raspberry Pi ™. This camera was implemented for image capturing. Although the Raspberry Pi ™ has more processing power than necessary for the project, its implementation is necessary for the testing that NASA has in mind for the DTN protocol. The plan is to utilize the Raspberry Pi's ™ extensive processing power in the future to handle an autonomous addition to the project.

*E. Propellers*

There is a tradeoff between propeller size and weight. The larger the propellers, the more they weigh and there is a corresponding decrease of flight time due to extra weight. The advantage of larger propellers lies in a faster response time and more thrust per revolution of each motor. Four 10 inch propellers were chosen to give the quadcopter the appropriate thrust power and maneuverability for its size and weight.

*F. Motors*

After some calculation using a flight time algorithm, a 900kV/200W motor yielded the best results compared to 800KV/300W and 1000KV/235W motors. For this project, the NTM Prop Drive Series 28-30S 900kv was chosen because it draws enough current and supplies enough thrust to give the quad-copter the desired agility and mobility, as well as optimizing the flight time and cost.

*G. Electronic Speed controller*

Electronic speed controllers (ESCs) do the job of taking a PWM signal from the microcontroller and converting it to the current the motors need in order to scale the thrust output. The Afro ESC Multi-rotor Motor Speed Controller was chosen based on its low price and slim shape. These ESCs can source up to 20 amps, which is appropriately more than needed for the motors used in this project.

*H. Frame*

Obviously, the quadcopter needs a structure for all of the on board components to be attached. When in actual flight the frame must withstand the environment, and affects speed, control, and maneuverability based on rotor separation. A relatively inexpensive fiberglass frame with dimensions (450mm x 450mm x 55 mm) was chosen due to its robust and lightweight (1.04 lbs) nature. The frame also includes motor mount bolt holes preset for ease of implementation. A picture of the actual frame in a primitive prototype phase is shown in figure 2.

**Figure 2: Quadcopter Test Unit**

*I. Battery*

Lastly, the batteries chosen were to have a high energy density and low weight overhead. Lithium polymer batteries provide this performance in a small size and weight package. Compared to traditional alkaline batteries, Li-Po batteries have a longer battery life and larger power capabilities while not being much larger or heavier. A single 3 cell 5000 mAh Li-Po battery pack was chosen because it was calculated to have the longest flight time while simultaneously providing sufficient power to the on-board components.

## III. Project Implementation

In order to fully comprehend the function of the prototype compared to the ultimate capabilities of this project, a distinction must be made between an intermediate model and a future version of the project. The following discussion compares and contrasts an ideal version of the project versus the current most sophisticated functioning prototype of the project.

The current prototype includes two quadcopters which can be manually controlled using two RC controllers. There also exists a ground station laptop having a custom designed graphical user interface (GUI) which sends commands to each quadcopter over a wireless network. Using the DTN software, the two quadcopters and the ground station create a "three corner mesh" network. As seen in figure 3, the two quadcopters and the ground station laptop represent three nodes in a three corner mesh network. Every node in the network is able to communicate with one another directly or is able to communicate with a node through another node. For example, if a first quadcopter loses connection with the ground station laptop, the first quadcopter can use the second quadcopter as a gateway to send and receive information to and from the ground station laptop.

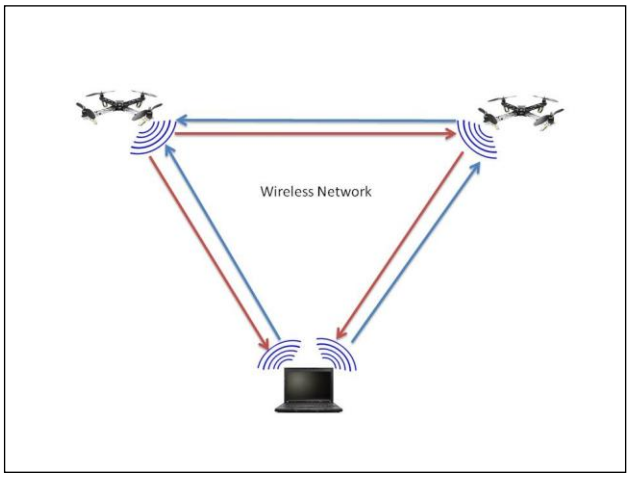


**Figure 3: Three Corner Mesh**

The function of the current quadcopter prototype is as follows. Each quadcopter sends the ground station laptop many different pieces of information in real time. The ground station laptop displays on its GUI both quadcopters' current GPS coordinates, altitude, battery voltage, the signal strength in percentage, the state of the quadcopter, and the wireless packet received number. The states of the quadcopter include hover, flying, landing, and idle. The ground station laptop also sends information to the quadcopter. The ground station laptop GUI includes buttons which command the on-board Raspberry Pi camera on each quadcopter to take a picture.

In this temporary intermittent phase, three team members are needed to accomplish the image mapping goal using two quadcopters. The first and second team members are each in charge of manual flight of one quadcopter, and the third team member is in charge of controlling the ground station laptop. Each team member controlling flight uses an RC controller to fly a quadcopter to a specific GPS coordinate and altitude. From there said team member hovers the quadcopter at the location with as little movement as possible while the ground station laptop controller team member commands the each quadcopter to take a picture, a single picture imaging only a portion of the total area to be imaged. The team repeats this process in multiple locations to obtain a number of aerial images.

After taking a sufficient amount of pictures in specific locations, the team uses an image stitching software to combine all of the pictures into a single photograph. It is worthwhile to explain that only one quadcopter is necessary to realize the goal of imaging an area. The use

of two quadcopters provides the ability to image a larger area in a shorter amount of time because both quadcopters can be in the air at the same time, each quadcopter taking only half of the total amount of pictures needed to image a given area; if one quadcopter is used to image the same area, one quadcopter would be in charge of taking all of the pictures, and the team may likely need to land the single quadcopter, switch out a depleted battery with a fully charged battery, and resume the flight pattern. A flight pattern of a single quadcopter is shown in figure 4 when the area to image is a rectangular area defined by coordinates 1, 2, 3, and 4.

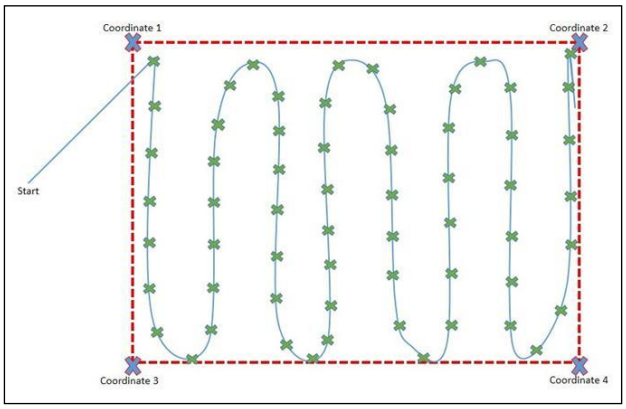


**Figure 4: Area Imaging Example**

The future goal is to have at least two autonomous quadcopters working in tandem to image a predefined area. The two quadcopters are to have the ability to "know" one another's location, and the system is to have software which makes sure that the two quadcopters do not collide with one another. Although autonomous flight is yet to be integrated, this feature has been in considered throughout the design, and a discussion of how autonomous flight control functions is in order.

In this future embodiment, only one team member is required to complete the mission. A single team member will simply input to the ground station laptop three or more GPS coordinates which together create an area for the quadcopters to image. From there the software takes the area and creates a multitude of "way points," each way point having a GPS location at which an image is needed to be taken. From there the software picks which way points each quadcopter is to handle. The software then creates an optimum flight path for each quadcopter. During a flight path, a quadcopter flies to, hovers, and takes pictures at each of its designated way points. The autonomous flight will use a software algorithm which compares the quadcopter's current GPS coordinates with the GPS coordinates of the desired way point and creates a directional vector pointing toward the desired location. The control algorithm then converts this directional vector into necessary thrust, pitch, roll, and yaw commands to be sent to the ESCs. The quacopter will compare the current location and desired GPS coordinates until they are identical. At this point the software recognizes the quadcopter is at a way point, it hovers, and takes a picture before moving on to the next way point. After a quadcopter takes pictures at all of its way points, the quadcopter flies to and safely lands at a predefined home location.

The task of autonomous flight is most definitely feasible with some time. The project required significant focus and time to create and perfect a custom PID control system for stability, so much so that additional time is required to implement autonomous behavior. The current prototype has all of the hardware and communication requirements required for autonomous flight such as an ability to send

and receive GPS coordinates, magnetometer data, and altitude data to and from the ground station laptop. After a thorough comparison of the prototype versus an ultimate version, an in depth discussion of how the components of the system interact is useful.

## IV. Hardware Design

A discussion is included which advances understanding of the way in which the components interact with the system as a whole in terms of communication and data transfer. Additionally, a discussion of the details concerning the PCB's schematic design is appropriate in the hardware design.

### *A. Communication*

The system transfers data to and from multiple devices for an explicit purpose. A discussion concerning the power distribution and important hardware communication lines is provided below.

1) RC Controller Receiver to Microcontroller

When purchased, the RC controller module used (a Turnigy TGY 9X) included a receiver therefore the project did not require any engineering development in order to receive RC controller signals. In contrast, the team needed to consider what signal lines are needed from the receiver to the microcontroller. Obviously, if and when autonomous flight is integrated, the RC controller would not be necessary but will still be integrated as a safety feature, the system containing a manual control override in case of an emergency situation.

The RC controller receiver is powered with 5 volts so a 5 volt power line and a ground line are connected from the

ESCs. Conveniently the microcontroller launchpad has a 5 volt pin and ground pin available for prototyping, and a 5 volt regulator is included in the PCB design. There are also 6 signal lines connected between the two devices. The first four lines send PWM signals for the thrust, roll, pitch, and yaw and correspond to left to right and up to down locations of the RC controller's two joysticks. These four lines transmit PWM signals to four GPIO pins on the microcontroller set up with interrupts which sample the signal periodically in time. The other two signals are used to read the position of a switch and the position of a dial on the RC controller. The team programmed the switch to "arm" the quadcopter. When in "unarmed" state, the switch effectively disables the motors by giving a pulse width corresponding to zero throttle to the microcontroller. This security feature also has an LED on the microcontroller which turns from a blue color to an orange color when the arm switch flips to an on state to signify the motors are enabled. The last and final signal from the RC controller to the microcontroller relays the position of a dial, the dial programmed to vary each of the three PID control gains during PID control system tuning.

2) ESCs to Motors

The electronic speed controller takes a DC current input and a PWM signal from the microcontroller and creates an AC current output. Specifically, each of the three output wires from the ESC to each motor carry an alternating trapezoidal wave at different phases compared to each other at any given time. Conveniently, each ESC outputs three female 4mm bullet connectors and each motor inputs three male 4mm with corresponding yellow, black, and red wires. One may be inclined to believe the electronic speed controller controls the speed of the motor by varying voltage or current, but surprisingly the speed is controlled by varying the timing of these three phases with respect to one another.

3) Microcontroller and Raspberry Pi ™ communication

The interaction between the microcontroller and the Raspberry Pi ™ is paramount in that the Raspberry Pi ™ is the device containing the Wi-Fi adapter to communicate with the ground station laptop. The microcontroller and the Raspberry Pi ™ use UART for data transmission. The GPS coordinates, altitude data, battery level, and state data read in or calculated on the microcontroller are combined into a text file, transmitted directly to the Raspberry Pi ™, then transmitted to the ground station laptop where the information is parsed and displayed on the ground station laptop GUI.

*B. Board Design*

The schematic layout as well as the PCB layout was designed in a Cad-Soft program known as EAGLE 6.2.0. This was a readily available program both as a free open version for student applications as well as a full version available on UCF engineering computers. This program provides a user friendly view of the interconnections on the board before routing and gives an ability to set up error criteria preventing a greater than 45 degree angle on any of the trace lines within the board.

The schematic design brought its own issues including implementation of parts that were not previously logged in the available parts library, the IMU components in particular. After the missing parts were created in custom developed library, the PCB design moved forward very smoothly without a hiccup. Then the decision of power ratings for parts came into play. When it came to assembling components onto the board, the team decided to use larger sized surface mounts (using size 1206 surface mounts instead of size 0402 for passive components with the exception of picofarad level capacitors which require a smaller sized surface mount) to make the soldering process easier. When it came to getting outside signals onto the PCB, the GPS connector is the only component which needed its own custom connector. Every other component is integrated on the board design.

In addition to the required signals for this project, certain additional circuitry and safety features were added. The first extra circuit gives the system an ability to do a quick reset through the use of a switch. This is included in order to avoid the need to pull power completely if and when the system gets locked up during testing. It was also decided that the PCB should not need to rely on the ESCs for power. This independent power capability is useful in PCB testing and is achieved through the use of a power jumper allowing for the +5V power line to come from either the ESCs (providing power during flight) or from the battery headers.

The decision was also made to protect the Lithium Polymer battery from receiving reverse current on its output. This battery protection is accomplished through the use of a diode from the battery to the main circuit. For the system to determine the state of the battery in each quadcopoter, a battery voltage circuit is included. This circuit implemented on the PCB includes a voltage divider circuit which reduces the battery voltage and sends the stepped to the A/D pin on the microcontroller to be read for telemetry data. The voltage must be stepped down due to of the voltage limitations on the A/D pins on the microcontroller. The pins themselves weren't designed to handle voltages much higher than 3.3 Volts. The microcontroller converts the voltage into a binary number which is converted through code programmed on the microcontroller into a percentage of battery life.

This team decided that the PCB should be smaller than the copter's platform (3.5" x 3.5") to avoid interfering with system integration at a mechanical level. With this criteria as well as the quantity of signals received by the central component, the Tiva C ™ microcontroller, required the PCB to need three layers.

After the design was laid out for the PCB, the chosen manufacturer had certain criteria to fill. Most importantly, the manufacturing of this board needed to be as economical as possible. Perceptibly, the company must be able to manufacture a three layer board. Additionally, the production and shipping of this board needed to be within three weeks of design submittal. OSH Park was the best company to fit the requirements at the lowest cost, and the actual unassembled PCB is shown in figure 5.

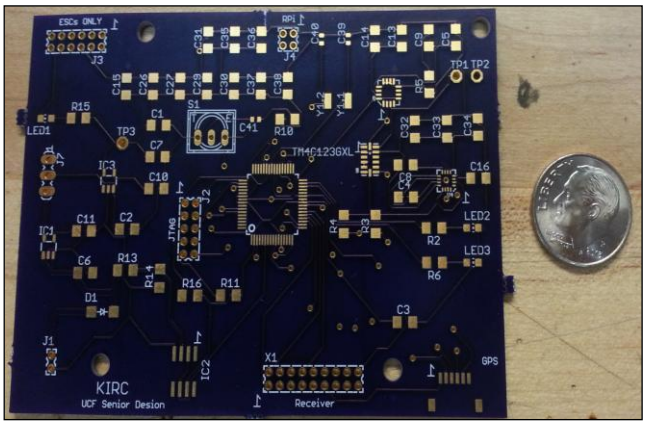


**Figure 5: PCB (Not Populated)**

## V. Software Design

This section includes information to aid in understanding the functioning of the control system and the real time operating system (RTOS). Specifically the Ziegler Nicole's method is used to tune the gains for the proportional, integral, and derivative terms in the control system algorithm. Discussion of the RTOS, employed in the microcontroller is presented first.

### *A. RTOS*

The reason the project needs an RTOS is so it can read all of the sensors in an efficient manor without tying up the processor and waiting for a response. The most important features of the RTOS are scheduling, priorities, and preemption. All of the tasks will have a priority assigned to them on the RTOS.

A pyramid of the peripheral priorities is shown in figure 6. The accelerometer and gyroscope are most important peripheral sensors because they are the main components used in the stability algorithm. The ability to read the accelerometer and gyroscope sensors is vital for the quadcopter to remain stable, and takes priority above all other sensors. The RC controller receiver is the next most important peripheral sensor because it receives commands from a team member manually controlling the quadcopter. Next in the hierarchy, listed from most important to least important peripherals are the altimeter, the magnetometer, the Raspberry Pi ™, and lastly the GPS.

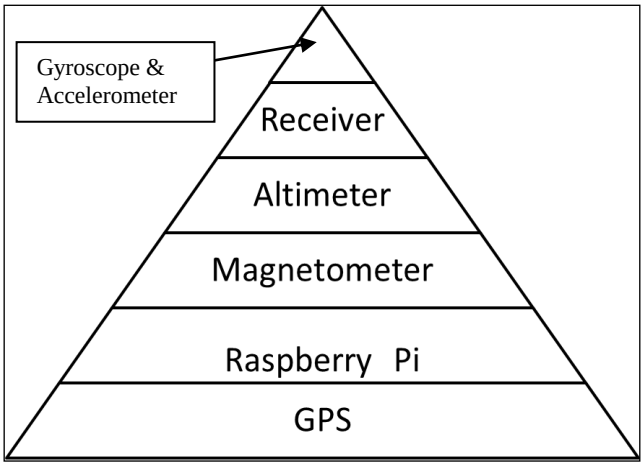


**Figure 6: Flight Computer Task Priorities**

Preemption allows the RTOS to remove a process while it is being processed. This is useful when a process gets stuck, a peripheral stops responding, or when something more important needs to be processed immediately. A process can get stuck for many reasons. For example, if the code being executed enters an infinite loop, the processor would be held up processing the infinite loop and no other processes could run. If the processor does not read its important sensors, such as the gyroscope and accelerometer, the quadcopter could lose control and crash. With preemption, the RTOS only allows the process to execute for a certain amount of time. If the process is not complete when the time is up, it will remove the process and allow the next scheduled task to execute. This would prevent one task from stopping all of the other tasks that need to be executed on time.

Programming for the microcontroller was implemented in Code Composer Studio ™ version 5.5. When the code for the microcontroller is compiled with the Code Composer Studio ™ Integrated Development Environment (CCS IDE), only the components of the RTOS that were used will be complied with the application to save memory on the microcontroller. Figure 7, reprinted with permission from Texas Instruments, provides some insight on how the code for this project is going to work with the TI-RTOS. [1] Everything that needs to be used for the quadcopter application will be declared in main( ). This

does not actually execute until the BIOS is called and everything is declared before that call.

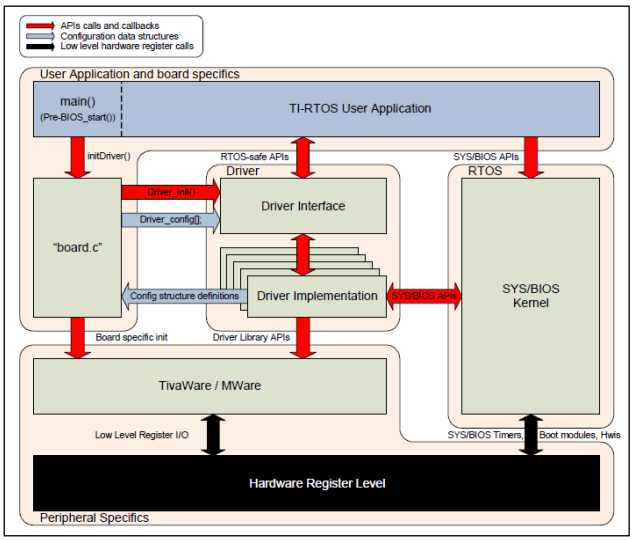


**Figure 7: TI-RTOS Overview**

The drivers that will be used for this application such as I2C and UART will be declared in the main function. This allows the CCS IDE to only compile the drivers that are needed. When the application wants to use a driver it simply makes an API call to the RTOS which then queues the task to be executed. TivaWare provides the code interface for the drivers and the operating system for things such as UART, I2C, A/D, and PWM. Now that an overview of the RTOS has been established, a discussion of the control system is appropriate.

*B. Control System Software*

The input of the control system of this project can come from two sources, either the autonomous guidance system, or the RC controller. Either the input can throttle the motors up, tilt the quadcopter in any direction, or spin the quadcopter. Using these methods, the quadcopters can fly and maneuver.

The fundamental control system of the project is shown below in figure 7. Control system uses the IMU to achieve feedback, while using a quaternion state estimator to fuse the information from the gyro, magnetometer, and the accelerometer. The quaternion state estimate can then be converted to euler angles to describe roll, pitch, and yaw. Using this method, the control system can be used to achieve attitude stability as well as control.

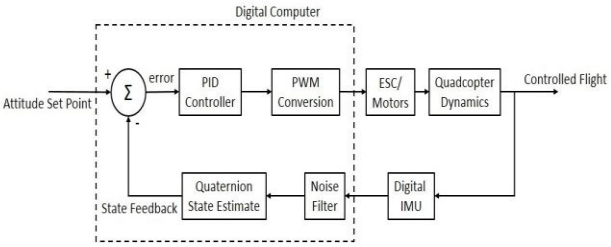


**Figure 8: Feedback Control System**

The RC transmitter/receiver unit on the quadcopter has multiple channels that can each represents an input to the system such as roll, pitch, yaw, throttle. Each of these channels can be used to control a flight parameter of the system. Using this as the input, the flight computer can subtract it from the feedback and obtain an error function that it can use as the compensator input. This can be expressed as

$$e(kT) = r(kT) - y(kT) \quad (1)$$

with r(kT) as the input, y(kT) as the feedback, and e(kT)as the error. The compensator of choice for this project is the PID controller, otherwise known as the Proportional Integral Derivative controller. This controller gets its name from exactly what it does to the input. The output equation shows the output equaling the sum of three terms,

$$output = u(t) = K_p e(t) + K_I \int_0^t e(t)dt + K_D \frac{de(t)}{dt} \quad (2)$$

each term in the equation bringing some form of value to the compensator. With each term contributing as a whole to PID controller, the brunt of the design is to determine the gains: $K_p$, $K_I$, and $K_d$ terms of the controller. The Ziegler-Nichols method is a well-known method of finding the three parameters based on trial and error. This method, requires patience and time. The trial and error process goes through several iterations before finally solving for all the parameters. This method also requires the use of a test setup where the setup isolates one axis at a time for tuning.

In order to tune the quadcopter, the group first isolated the axis for pitch, set the proportional and integral gains equal to zero, found $K_u$ and $T_u$, then calculated the initial $K_p$, $K_I$, and $K_d$. Realizing these values will not yield perfect stability, the team set the dial on the RC controller to vary the proportional gain over an isolated range, the range including the calculated proportional gain $K_p$. The team turned the dial until the quadcopter seemed to have minimal oscillations. The team repeated this process to tune $K_I$, and $K_d$. It should be noted that as soon as one

gain was manipulated, the system behaved differently in terms of stability which required re-tuning of the other gains yet again. Eventually the team agreed upon 3 values for $K_p$, $K_I$, and $K_d$ which were observed to yield a responsive and non-oscillating stable quadcopter on the respective axis. Due to symmetry, the pitch and the roll gains should identical but the process still is repeated, isolating the yaw axis. After the team decided on values for these axes, the team needed to again tune the gains during flight. In summary, the team used the Ziegler Nichols trial and error method to tune the gains of the custom PID control system, and the team implemented a real time operating system to prioritize the tasks and assure the system does not fail, which would be catastrophic.

## VII. Conclusion

In conclusion, the resultant project yields two manually controlled quadcopters capable of wireless communication with a ground station laptop. Designing the PID control system algorithm from scratch gave the project a higher level of difficulty. The Ziegler Nichols method to obtain the proportional, integral, and derivative gains required significant time and effort to test and tune these parameters to perfection. Although the quadcopters are not yet autonomous, the custom control system for the project is specifically implemented to give the project an ability to incorporate this feature in a future prototype.

The quadcopters and the ground station laptop send and receive information over a delay tolerant network embedded on the Raspberry Pi's Linux platform. The ground station laptop has a GUI which displays both quadcopter's GPS coordinates, altitude, state, and battery life in real time. The GUI also had buttons which can take a picture on either one of the two quadcopters at any given point in time with an end goal to image a predefined area.

## Acknowledgement

The authors wish to acknowledge the assistance of NASA for their generous budget contribution made possible by Marc A. Seibert, the supervisor of the Co-op student working for the Center Planning and Development Office at Kennedy Space Center. This project would not be possible without his direction and support.

The authors also wish to recognize Anita Jones from Lockheed Martin's Missiles and Fire Control facility who aided in soldering difficult components on to the PCB and Dr. Samuel Richie for making this senior design project such a valuable learning experience. Last but certainly not least, the authors would like to thank Dr. Chan, Dr. Wasfy, and Dr. Yuan for not only taking time out of their busy schedules to review our project but also for giving us inspiration and invaluable knowledge throughout our engineering curriculum.

## The Engineers

**Nathaniel Cain** is an Electrical Engineering senior at UCF. Currently, he works at NASA's Kennedy Space Center through the pathways internship program. After graduation, he hopes to continue working for NASA long term. His primary interests include hands on projects with avionics and electrical systems.

**James Donegan**, an Electrical Engineering major enjoys spending his free time playing indoor and beach volleyball. He interns at a startup engineering firm called Unikey Technologies Inc. and works as an assistant for an independent patent attorney. He aspires to go to law school and become a patent attorney.

**James Gregory** is an Electrical Engineering major from Jacksonville, Fl. James currently interns at Lockheed Martin through the college work experience program and recently accepted a full time position as an electrical engineering associate.

**Wade Henderson**, a Computer Engineering major, has interests in computers and electrical circuits. During his free time he likes to build electronic devices. Wade started an internship with Unikey Technologies Inc. last May and will pursue a career in firmware development.

## References

[1] Texas Instruments Inc., "TI RTOS User Guide," Version 1.21, Page 41. Retrieved 30 March 2014, Website:<http://www.ti.com/lit/ug/spruhd4e/spruhd4e.pdf >

[2] Ziegler, J.G and Nichols, N. B, "Optimum settings for automatic controllers," 1942, pp. 759-768.

[3] Franklin, G. F. et al. "Digital Control of Dynamic Systems," section 3.3: PID Control, 1998, pg 54