MicroCART May14-10

Design Document V1

Team Members:

Kevin Engel

Nathan Ferris

William Franey

Mike Johnson

Kelsey Moore

Lucas Mulkey

Aaron Peterson

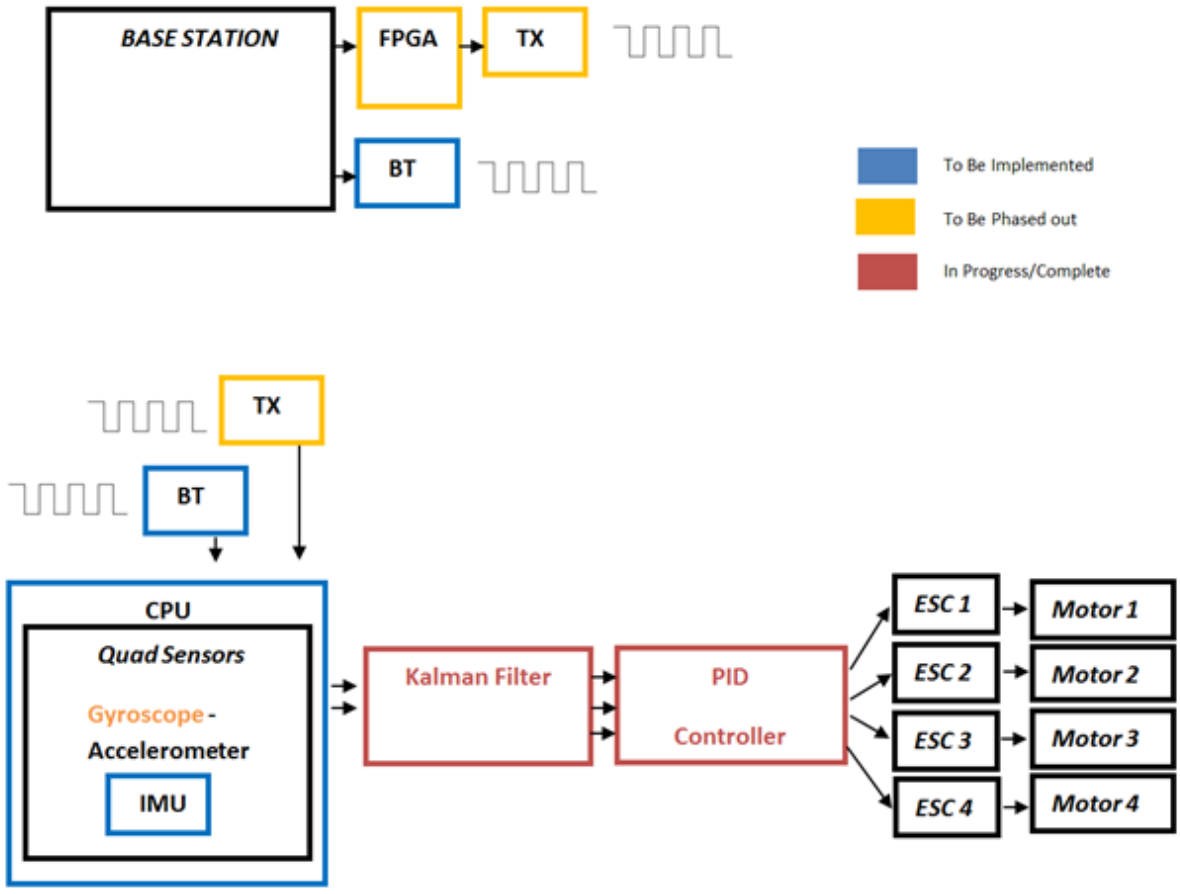
Advisors:

Dr. Nicola Elia

Dr. Phillip Jones

Table of Contents

System Design



Functional Requirements

Control via manual controller or base station GUI

Non-Functional Requirements

Safety and Visually Appealing

System Requirements

The current MicroCART system is only functional indoors and is heavily reliant on an expensive high-speed camera system. This severely limits mobility and reduces potential for real-world applications. The current mission is to convert the system from

being completely controlled by an external computer and camera system, to one that is almost entirely onboard the UAV. This will require using Bluetooth to communicate between a laptop base station and the on-board sensors and FPGA. These changes will allow us to fly outside and demonstrate some autonomous functionality.

System Integration

Commands are sent from the ground computer to the FPGA in a formatted array of data. The FPGA interprets the data array and generates a PPM with a channel for throttle, roll, pitch, yaw. This output from the FPGA is the input to the transmitter, which wirelessly transmits the new quad values to the receiver on the quad. The quad's receiver converts the PPM signal into separate PWM signals and forwards these PWM's to the GU344 controller. The GU344 controller analyzes the PWM signals and controls the motor speeds appropriately. The orientation and position of the quad is tracked by the OptiTrack camera system. This information is communicated to the ground computer via the VRPN protocol. The ground computer's PID controller compares the measured vs expected position and orientation, and computes the error. The computer then sends an new command with intent to fix this error, and the cycle repeats.

Detailed Design

Hardware Details

I. Base Station

The base station PC is used to interpret data from the camera system and send that information to the FPGA, as well as take user commands to control the quad. Currently, it has both a low-level terminal-based interface and a full graphical interface.

Inputs: Camera System Data, User Commands

Outputs: Pitch-Roll-Yaw data to the FPGA

Interfaces: GUI, FPGA Board

II. Base Station FPGA

The base station FPGA is the intermediate between the base station PC and the quad. It simply takes in the PCs byte-array output (pitch, roll, yaw) and sends them to the quad as PPM signals.

Inputs: Pitch-Roll-Yaw Data from PC

Outputs: PPM Signals to Quad

III. High-Speed Camera System

The high speed camera system is our current method of orientation. This system tracks four small infrared reflectors located on top of the quadcopter and quantifies the information (x,y,z) and outputs via the VRPN protocol to the base station.

Inputs: Location of infrared balls

Outputs: Position and orientation data to the base station via VRPN

Interface: OptiTrack Software

IV. LABET FPGA

The Labet FPGA board, located on the quadcopter, will be used to stabilize and control the quadcopter. Currently, this board is off the vehicle while we experiment with wireless communication.

On the board: IMU unit (3-axis gyroscope, magnetometer, accelerometer), BT module, GPS module, VHDL code.

Inputs: Orientation data from the camera system and base station or the IMU.

Outputs: PWM signal to quads motors.

V. Quadcopter

The quadcopter receives signals through the receiver in the form of PPMs. The PPMs are sent to the Spektrum receiver where they are converted to PWMs for each of the following directions: throttle, pitch, roll, yaw. Each of the signals goes through the GU-344 where the internal accelerometers help balance the outputs to the motors. These signals then travel to the ESCs, which tell the motors how fast to spin. Once the PWM VHDL modules have been completed, the GU-344 will be taken away since its internal accelerometers have drifted over time and give inaccurate readings. A new IMU will be placed on the quadcopter and provide the data that was previously provided by the camera system.

The ESCs connect directly to the battery for power, and the motor connects through the ESCs. The ESCs determine how much voltage to send to the motors

by the duty cycle of the PWM. The higher the voltage to the motor, the higher the revolutions per minute (RPM).

On the copter: Speed Controller x4, Motor x4, FPGA Board, GU-344(to be phased out), LABET Board, 2200mAh Battery

Inputs: PPM signal from base station

Outputs: PWM signal to ESCs

VI. Wireless Communication

To utilize wireless communication functionality, we have multiple XBEE Bluetooth modules that communicate through UART. As such, they can be written to and read from as file descriptors in a C program. Currently, we are using these modules to obtain real IMU data from the quadcopter, with which we can use to test our Kalman filters.

VII. GPS

Eventually, we plan to remove the camera system from the project, and use onboard GPS for location. This presents an obvious design challenge, as the GPS is both less accurate and far slower than the expensive high-speed camera system.

Module- GPS Bee with onboard mini-antenna

Position Update Rate - 4Hz

Interface - UART

Software Details

I. Camera Sensor Data

The camera system data is used to determine the orientation and position of the quadcopter. The quad can receive this data through the Virtual-Reality Peripheral Network (VRPN) classes provided to work with the camera system. These classes abstract all of the detected object by the camera system, and provide specific metrics like position (Cartesian coordinates) and orientation (pitch, roll, yaw, heading). The PIDs and base station need this data to stabilize the quad and provide the user with up-to-date information on the quad's position.

II. Onboard Sensor Data

The onboard sensor data will be used as the camera system is phased out. The data from the IMU will be used to determine orientation data in place of the cameras, and the GPS will be used to determine location data in place of the cameras. Both of these will enable the quadcopter to fly in an environment outside of the lab. The IMU data will be interpreted in the C code onboard the quadcopter, and then it will be given to the Kalman filter. The Kalman filter will give this data to the PIDs to stabilize the quad.

III. Kalman Filters

The Kalman Filter takes the data from our onboard sensors and combines it all together in an attempt to limit the shortcomings of their independent uses. The accelerometer keeps the error within a limited range while the gyroscope accurately depicts the trend of the quad's attitude. This filtered measurement gives the best estimate of the quad's current location and sends this data to the PIDs.

IV. PID Controller

The PID controller looks at the current position of the quad and tells the quad how to get to another location. It calculates the the difference between our desired location and our current location, called error, and adjusts itself according to the sensitivity of the P, I, and D terms that we design for the controller. Roll, pitch, yaw, throttle, x-position, y-position,and z-position all have their own specifically tuned gains that pass through the PID controller, which attempts to minimize the error term to reach the target location.

V. GUI

The GUI system is used to interact with the quadcopter using the Optitrack camera system, the base station FPGA board, and the manual controller. The GUI starts interacting with these by first taking in camera system data. It then does a quick Kalman filter and PID calculation with the information received.

Upon start of the GUI, the code implements a home system. It takes the x, y, and z coordinates that the quadcopter is currently at and saves it as the home location. From this location, the GUI directs the quad in the direction it needs to go in, adjusting the pitch, roll, and yaw to keep it in that position until it receives the command to start moving.

The GUI consists of six different parts. These are the Settings, the Grid, the Console, the Omni-bot, the Sensors, and the side bar. Our implementation doesn't use the Omni-bot section. For now, just know the Omni-bots are there and this section is used to direct whether the quad-helicopter follows the omnidirectional robot or whether it follows the standard commands given to it.

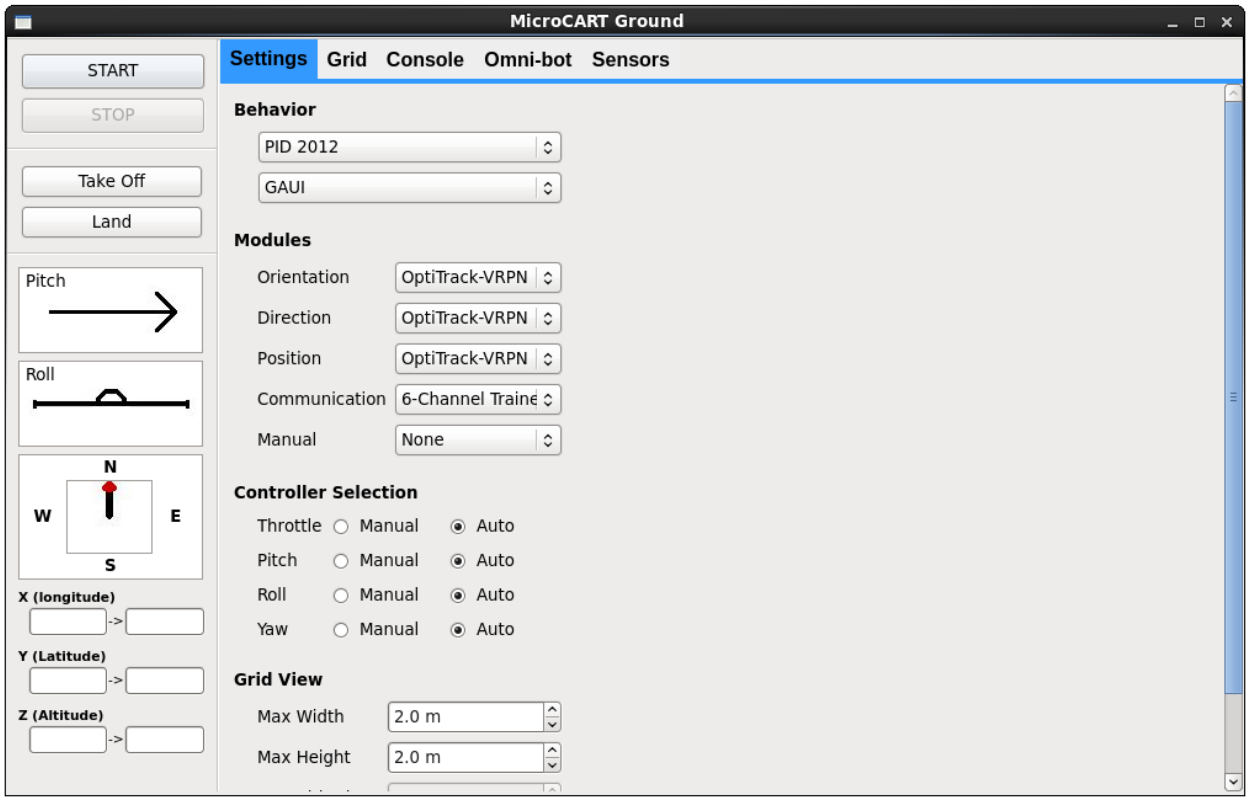
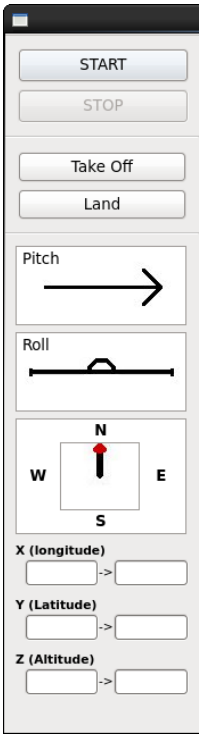


Figure 1: Overall look of the GUI when on the settings option.



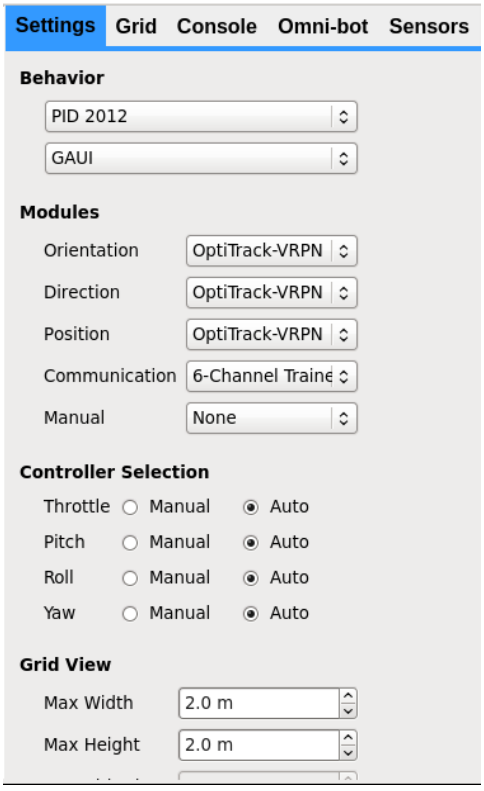
A. Side Bar

Upon initially starting up the GUI, it connects to the base station FPGA board. The image on the left shows the sidebar of the GUI that is static throughout each view.

The bottom of the sidebar contains the x, y, and z coordinates that the quad-helicopter is at in relation to the home position. The pitch and roll graphics in the center show the orientation the quad-helicopter is currently in. The compass shows the quad-helicopter's current heading in relation to the camera system.

The buttons toward the top of the sidebar show the quad-helicopter's takeoff and landing functionality as well as the ability to start or stop the communication link. When clicked, the START button begins communications with the camera system and initializes the PPMs sent over to the quad-helicopter. The STOP button stops the system from reading camera data and stops the transfer of data to the quad.

The Take Off and Land button are only functional after the START button has been pushed. The Take Off button is an auto take-off feature for the quad to get in the air on its own. It works within the code by slowly raising the throttle PPM signal sent through the base station FPGA board. The GUI also uses camera system data to keep the quad stable as it is raising off the ground. The auto-takeoff feature is currently set to raise the quadcopter until it gets 0.5 meters off the ground.



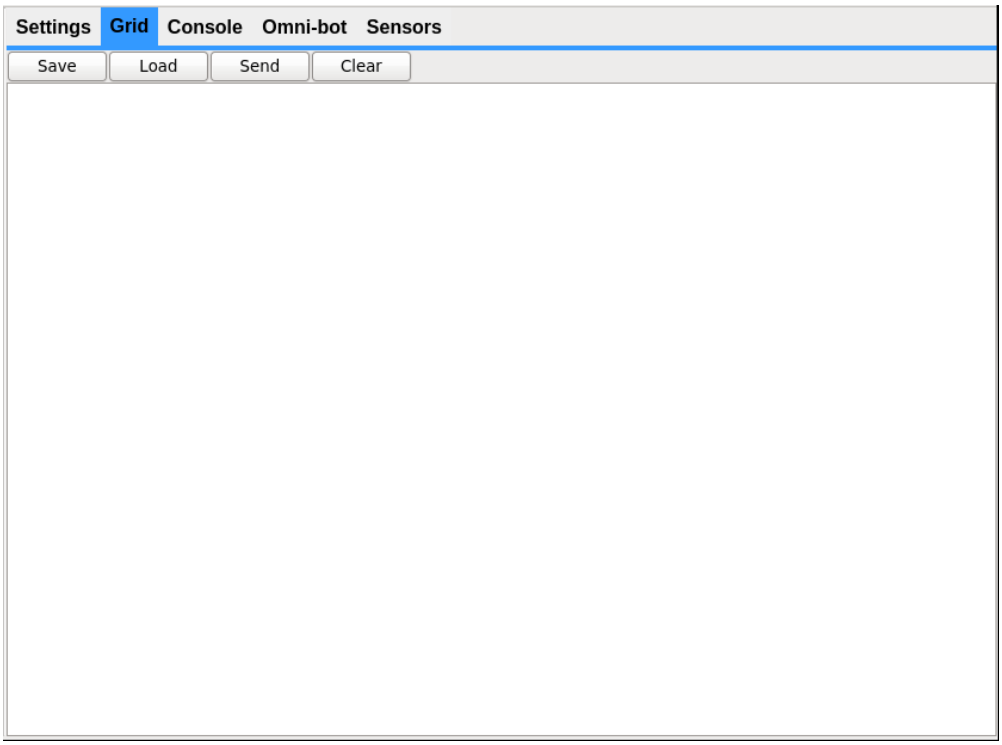
The Land button brings the quad onto the ground from the air. The GUI uses the quad's current position to keep it balanced while slowly reducing throttle. Once the quad gets to about 0.1 meters off the ground, the GUI shuts off all the motors at once so that the quad doesn't bounce upon hitting the ground.

B. Settings

The settings view in the GUI is pictured in Figure 3. This area is divided into the four following categories: Behavior, Modules, Controller Selection, and Grid View. The Behavior section is used to determine what system it will be running on. It currently only has two behaviors, PID 2012 and Test. The only system actually used to fly the quad-helicopter is the PID 2012. The other option is the GUI's visual style, but there is only one option at the moment.

The Modules and Controller Selection determine which controller will be used to fly the quad-helicopter. The default option is OptiTrack-VRPN, and that is currently the only option capable of running the flight controls. Communication can be set to the 6-channel trainer or the 4-channel trainer. The Manual control selection determines whether the quad is fully automatic, or controlled manually when manual control is selected. The Controller Selection allows you to change which specific controls will be either automatic or manual.

C. Grid



The Grid's basic functionality is to have the user draw a flight path for the quad-helicopter. The desired flight path is drawn in the white box using. Once the path is drawn, the user can click the send button to start the quad-helicopter's new flight path. When the user clicks the clear button, the drawn flight path is erased, allowing a new flight path to be created. The load button is used to load an existing flight path, and the save button is used to save the path to reuse later.

D. Console

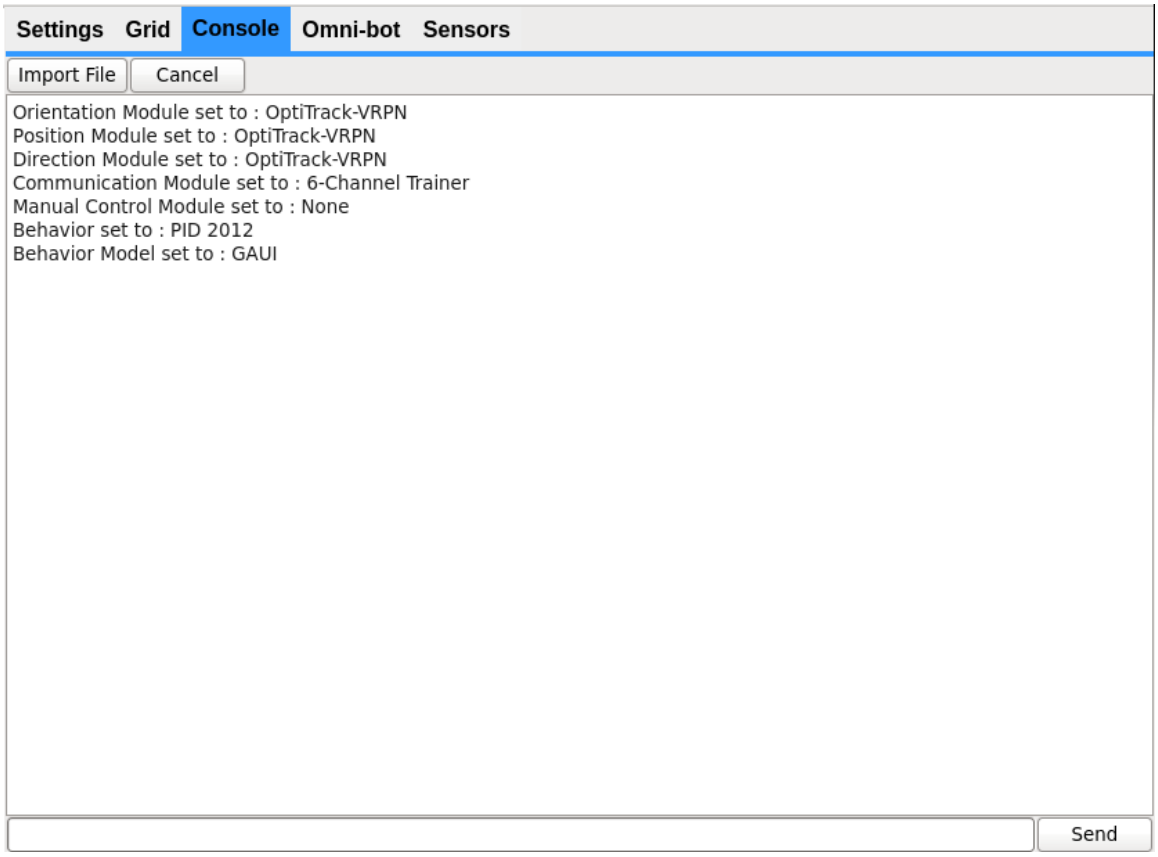
The console is used as a text-based controller for the quad-helicopter. The current supported commands are the following:

alt - used to increase or decrease altitude

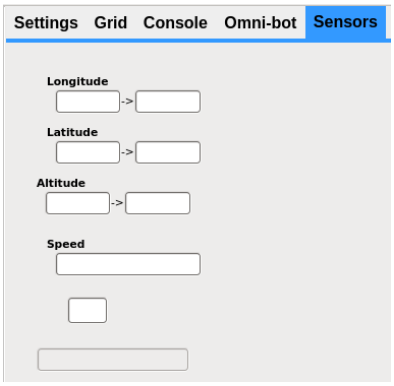
lon - used to increase or decrease longitude

lat - used to increase or decrease latitude

The console receives and outputs data given from the settings section in real-time and displays that data. The user types in the command to give toward the bottom of the interface and clicks the send button to initiate the command. The quad-helicopter measures all changes given from the console in meters.



E. Sensors



The Sensors area of the GUI is non-interactive. It shows where the quadcopter is going on the left side, and where the quadcopter is currently at on the left. The user can see the longitude, latitude, and altitude the quad is currently at and he or she can this the information for testing purposes. This information can also be used to make sure the

quad-helicopter is reading information correctly, ensuring the quad does not leave its designated area.

VI. CLI

The CLI (command line interface) will be a menu driven interface that is simpler and faster than the GUI. The supported commands will allow for incrementing and decrementing the altitude, latitude, and longitude for movement, as well as incrementing and decrementing throttle, pitch, roll, yaw for testing.

Test Specification

1.) Testing of motor and ESC performance:

i) attach IR reflecting tape to the motor, or in the case that no IR tape is available: aluminum foil and transparent tape may also work.

ii) connect a DC power supply of 10 volts, or whatever your battery supplies, to the battery connection of the ESC.

iii) The function generator will require 3.2 Volts, peak to peak with an offset of 1.6 Volts (or 1.6 Volts amplitude), square wave, with 280 hertz to the ESC inputs. Set the duty cycle to 28%. If setting limits for the motors, move to step iia, else move to step iii.

a)Set the upper limit with a duty cycle around 45%, but no higher than 50%. Turn on the power to the motor.

b) Once the motor beeps, lower the duty cycle to the lower limit requested.

iv) observe the rotation speed with respect to duty cycle using a tachometer.

2.) Testing of the PPM outputs from base station computer

i) turn on oscilloscope and attach positive and negative probes to positive and negative hooks on the PMOD attachment to the Digilent board; adjust the voltage and time scales to obtain a readable image of the PPMs.

ii) run the GUI or the CLI to send commands from the base station

iii) verify the commands have changed the PPM values accordingly. Order should be roll, pitch, yaw, throttle

Simulation/Modeling

Modeling the system consists of placing components together in order to model the system. Starting with the IMU, the inputs/outputs consisted of roll, pitch, and yaw data. From there, the outputs enter the Kalman filter. Here, the data eliminates noise and

corrects the drift. Leaving the Kalman filter, the refined data is pushed through the PID controller, which is where the stability criteria is calculated.

This processed data is accounted for in the base station, the modeled software (Kalman filter, PID controller) then sends the adjusted data back to the quad's receiver via PWM's. The filtering and PID corrections now run through the ESC's and are used to change the motor outputs to account for the imperfections in IMU data.

Once the software models are put in place and tested for functionality, the next step is to run through a simulation. To start, artificial data is used to test the Kalman filter and the PIDs. Upon acceptable results, simulations are started.

Simulations begin with turning on the IR camera system and checking for a response from the quad to the camera system. Safety checks consist of clearing all possible objects within the testing area, tightening of the restraining tether, and having a clear view from the driver to the quad. By repeating these checks before every flight, we achieved a consistent test pattern as well as maintained an unharmed quad and team.

Next was to attach a fully charged battery onto the quad, a set of four "beeps" indicates a correct connection. Assuming that the previous precautions are accounted for, the driver then uses the base station transmitter to manually lift the quad into the air. Once an altitude of approximately one meter is met, the autonomous switch is flipped, putting the quad into a stand-still position set by the driver. From here, simulations begin. One simulation that is very useful is gently "rocking" the quad in various directions to test and tune the PID coefficients. Camera system data can also be acquired when the quad is running during simulation. Take-off's and landings are also practiced via simulation, this step will show if certain motors or ESC's are underperforming.

By creating a routine simulation process which acquired various sets of data, the team can then account for this data in the models. Adjusting and running models is an efficient swift process for quickly making changes and tuning the system. However, models cannot take the place of actual simulations, which is why the safe and strict simulation rules were set in place to produce consistent and precise data for which to rely on.

Software Tools

1. Eclipse

   Purpose: IDE to write and execute programs on the LABET FPGA board (ex. sending binary data, reading IMU data).

   Setup: *See Quartus setup*.

2. Quartus

   Purpose: Software tool to program the LABET FPGA hardware with a hardware description language (VHDL).

   Setup:

   http://wikis.ece.iastate.edu/microcart-senior-design/index.php/FPGA_Board

   under “Flying”, click on “FPGA Board”

   scroll down to “Compiling/Programming”

3. QT Creator

   Purpose: IDE to create the graphical user interface, and write the base station logic.

   Setup:

   terminal command: qtcreator &

4. ISE

   Purpose: IDE for writing, simulating, and generating bitfiles of VHDL code for the ground FPGA board. Use Adept software tool for programming the bitfiles to the Xilinx FPGA on the ground FPGA board from Digilent.

   Setup:

   http://class.ece.iastate.edu/cpre583/

   Under homework: Tools Overview (ppt)

Hardware Tools

1. Hyperion EOS1420i NET3 battery charging station

   Purpose: Charging the batteries that power the quad.

   Setup: Turn on power supply to charger. Make sure the voltage is correctly set, roughly 20V. Plug in the battery and the balancer. Memory 5 should be on the screen. Press and hold enter until the screen goes to a battery check. After the check, press and hold enter again. The screen should show the number of cells,

which is three in our case. Press enter one last time and the battery should begin charging. Once it is done charging, the charger will play a lyrical tune until the enter button is held or the power is turned off.

2. Futaba RealFlight Transmitter

   Purpose: Controlling the quad manually through the GUI.

   Setup: *See hard copy instructions for calibration and button functions.*

3. Spektrum DX6I Transmitter

   Purpose: Main controller for sending signals to the quads receiver. All controls eventually pass through this controller.

   Setup: *See hard copy instructions for calibration and button functions.*

References

Kim, Phil. *Kalman Filter for Beginners with MATLAB Examples*. Trans. Lynn Huh. Republic of Korea: A-JIN Publishing, 2011. Print.