5 Testing

Because our project involves many components, testing was first performed on each component to ensure they worked properly before testing the entire system:

|  |  |
| --- | --- |
| **Hardware Components** | **Test** |
| Motors/ESCs and Transmitter/Receiver | Confirm all motors spin by changing transmitter input |
| Power Distribution Board | Connect all ESCs to power distribution, power with single battery, and confirm all motors spin |
| 5V Switching Regulator | Confirm battery input was converted to 5V output |

As described in the hardware section, the drone system consists of a variety of components: a transmitter, a receiver, a gyro, a 5V regulator, 4 motors, and 4 electronic speed controllers (ESCs). The first test we performed was to make sure all of our hardware components functioned as intended. We tested each motor and ESC combo by powering it with a battery, connecting it to the receiver, and varying the throttle to check if the motor would spin. We repeated this test with each motor and confirmed that all 4 motors and ESCs functioned correctly. This test also established that the receiver and transmitter were functioning properly because the signal was transmitted successfully to make the motor spin. Next we tested the power distribution board by plugging in all four ESCs to the board, connecting each ESC to the receiver, powering the distribution board with a single battery, and confirming each motor would respond to transmitter input. Finally, we tested the 5V switching regulator to confirm it could convert the battery’s 11.1V to 5V by checking the output with a multimeter.

Once each component was tested, we moved on to begin completing the critical milestones necessary to complete our project.

|  |  |
| --- | --- |
| **Critical Milestones** | **Technical Description** |
| Read Gyro data | I2C communication from Gyro to FRDM K64 |
| Control Motors using FRDM K64 | Write PWM from FRDM K64 to ESCs |
| Interpret Receiver input using FRDM K64 | Read PWM from Receiver to FRDM K64 |
| Establish untethered Power | Power all gyroscope, FRDM K64, and Receiver using 5V regulator without USB input |
| Make Drone react to Transmitter | Transmitter input translates to appropriate drone movements in throttle, roll, pitch, and yaw |
| Make Drone React to Gyro | Gyro input translates to appropriate drone movements to eliminate angular velocity in roll, pitch, and yaw axes |

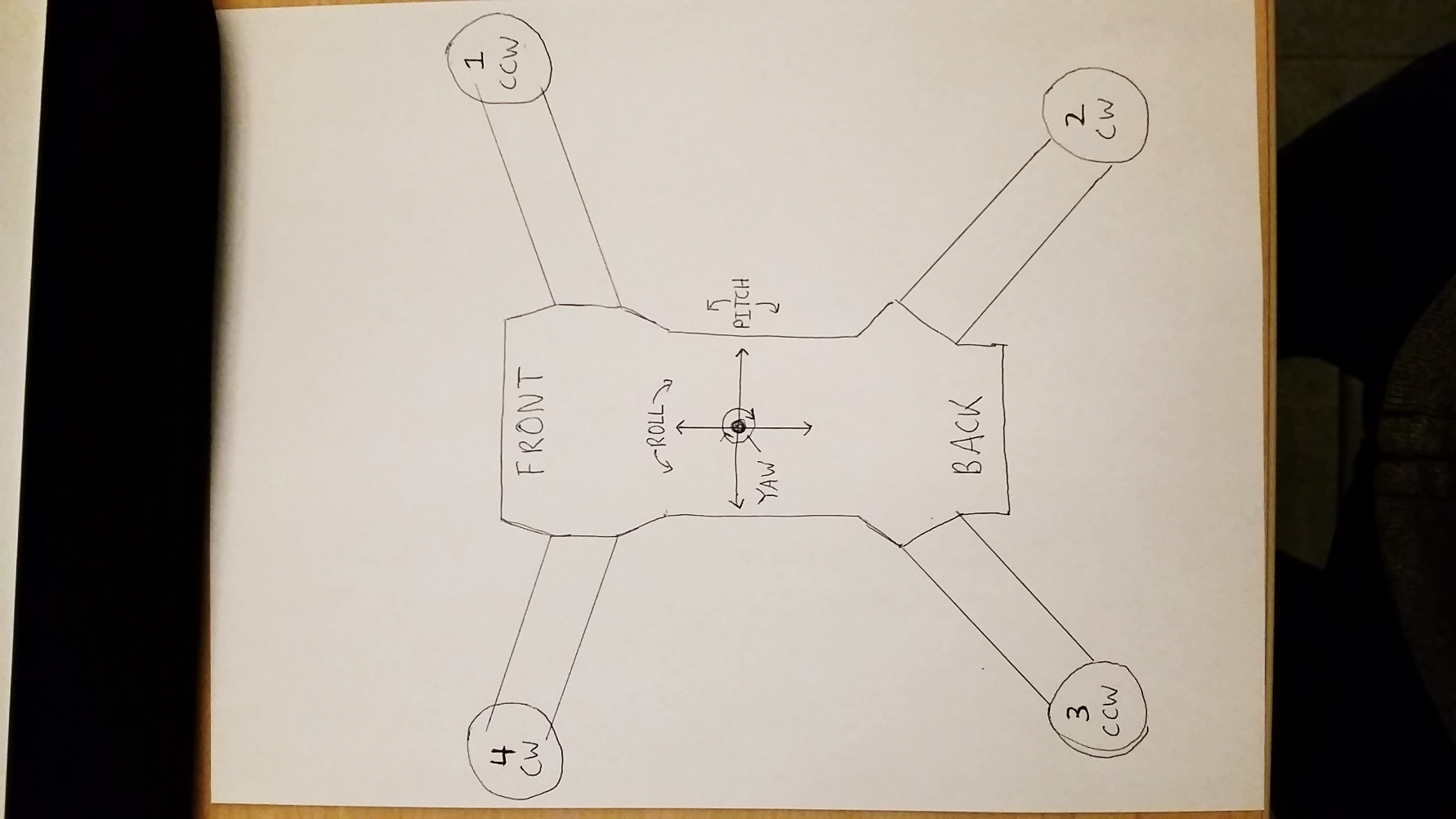
Our first milestone was to successfully read gyro data. First, we wired the gyroscope to the FRDM board using I2C for data transmission. Then we implemented and modified an I2C library found on mbed to communicate with the gyroscope. We set the appropriate addresses and registers necessary for communication. Then we used serial to read back the roll, pitch, and yaw values from the gyroscope. We used RealTerm, a Serial Terminal software, to establish a serial connection and monitor the data stream coming from the gyro. After some trouble shooting, we successfully read roll, pitch, and yaw, values from the gyro. Next, we tested to see if the data made sense. We would accelerate the gyro in one axis and check to see if that axis value increase. We did this for each axis, determining which direction of movement caused a positive and negative value. We recorded theses direction and noted which axis would be used for roll pitch and way based on the placement of the gyro on the drone.

Our next goal was to control the motors directly from the FRDM board via PWM without a transmitter and receiver. We implemented a PWMOut library to control the PWM input to each ESC. We connected each ESC ground wire to ground and signal wire to a PWM port on the FRDM K64. Then we created four PWMOut objects and set their pulse widths to 1500us. After we flashed the code, all motors successfully spun. We then tried varying the pulse width from high to low using a for loop to find the pulse widths corresponding to the motor’s maximum and minimum speeds. We found that each motor’s pulse width range was between 1 and 2ms.

Next, we worked to have the FRDM board read the PWM signals generated by the receiver. We implemented a PWMIn library to help us determine the pulse width using interrupts and timers to measure the time difference between the rising and falling edge of the signal. We tested this by using serial to print the pulse widths of each channel to RealTerm. To test if it was working correctly, we varied the transmitter inputs and confirmed each channel’s pulse width varied from 1 –2ms as the channel input was swept from minimum to maximum position. After scaling the values appropriately, each channel’s pulse width changed appropriately based on the transmitter input.

Because the drone is designed to fly, we could not rely on the USB cable as a permanent power source. The next test consisted of ensuring all electronic components (receiver, gyro, and FRDM board) could be powered via a lithium polymer battery. We referenced the FRDM board’s datasheet and found the board could be powered externally via the Vin GPIO port. But because the Vin port can only handle 5-9V, we used a 5V switching regulator to step down the voltage. We connected the battery to the regulator, and the regulator output to the Vin port on the FRDM board. The FRDM board successfully received power. Then we tried powering the Recover, which takes a 4-6V using the 5V GPIO port on the FRDM board. Unfortunately, the receiver would not turn on. Instead we directly powered the receiver using the 5V regulator and this worked successfully. Finally, we powered the gyroscope using the 3V3 GPIO port on the FRDM board. We plugged in the battery to the regulator and all components were successfully powered.

The next test was to confirm that the motors would respond to transmitter input. After successfully reading PWM from the receiver and writing PWM to the motors, this test was to ensure the transmitter inputs would result in the correct response by the drone. Because this test was done without propellers mounted to the motors, we monitored the pulse widths of each motor to tell its speed. We used serial to print the PWM values for each motor every 250 cycles of the main loop. This test was broken into four components, throttle, roll, pitch, and yaw. The expected motor results based on a given change in input is described in the tables below.



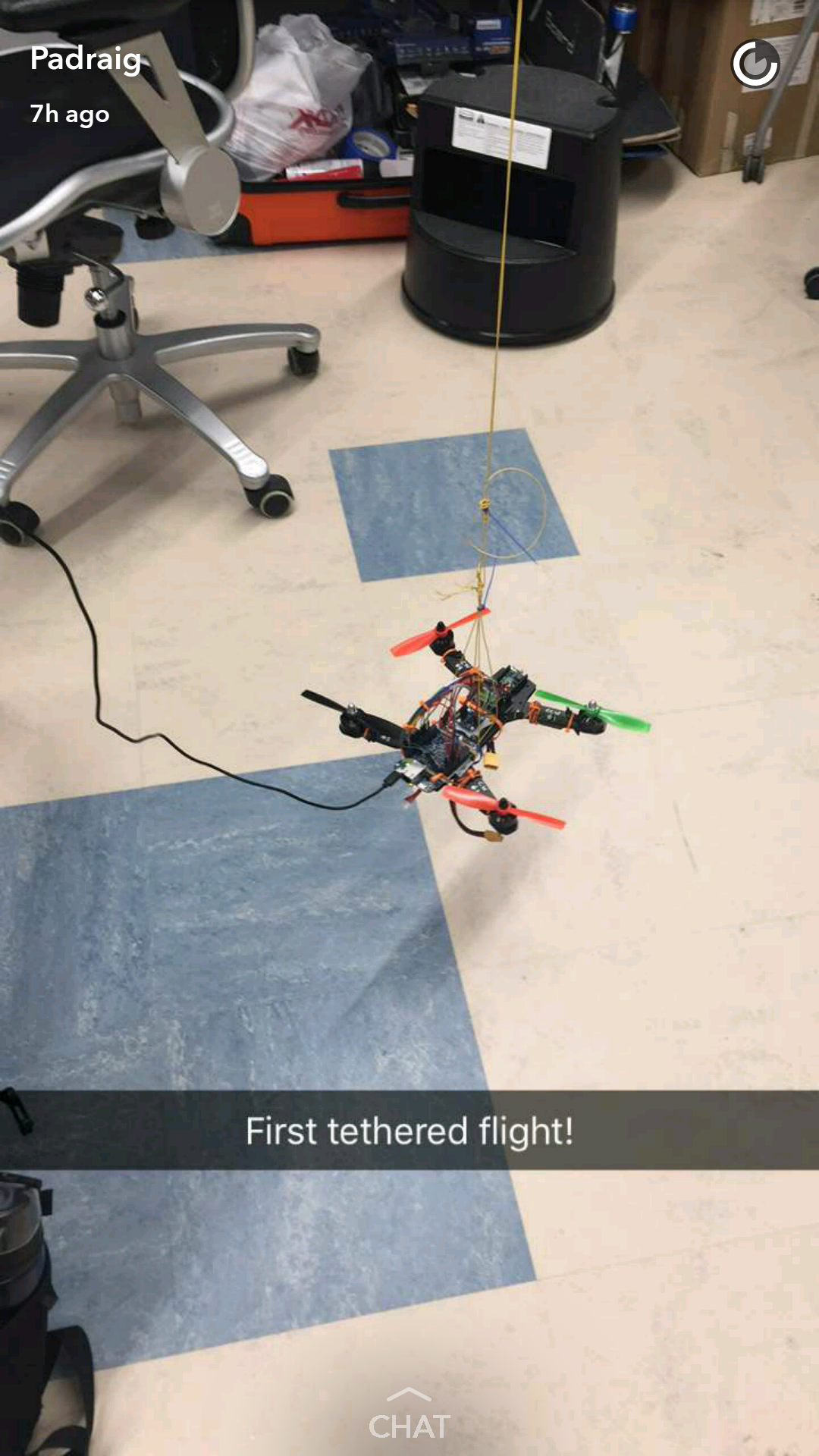
|  |  |  |
| --- | --- | --- |
| **Channel** | **Input** | **Motor Response** |
| Throttle | Increase | 1  2  3  4  1  2  3  4  1  2  3  4  1  2  3  4 |
| Decrease | 1  2  3  4  1  2  3  4  1  2  3  4  1  2  3  4  1  2  3  4  1  2  3  4  1  2  3  4 |
| Roll | Right | 1  2  3  4 |
| Left | 1  2  3  4 |
| Pitch | Forward | 1  2  3  4 |
| Backward | 1  2  3  4 |
| Yaw | Right | 1  2  3  4 |
| Left | 1  2  3  4 |

First the throttle was swept form high to low to ensure all four motors simultaneously changed speeds. Then yaw was tested by moving the yaw channel right and left and confirming the appropriate motors would speed up or slow down. This was done for each channel delineated in the table and the results were confirmed by monitoring the PWM values of each motor. After changing some of the signs in the code, we achieved all the desired motor responses.

The final critical milestone to complete before beginning full system testing was to ensure the gyro could enact the correct motor responses. The purpose of the gyro is to reduce the error between the input (setpoint) and current position. When the input is untouched, all setpoints are set to zero, and the drone’s angular rate in all axes should be zero. If the input is not zero, for example, the roll is set to the right, the drone’s angular rate should match the setpoint of the transmitter input. In other words, the drone will continue rolling at the given rate until the input is set to zero again. One method for testing if the gyro is enacting the correct motor responses is to set the transmitter inputs to zero, physically rotate the drone in a specific axis, and see whether or not the motors respond trying to cancel the angular rate back to zero. Again, because this test was not done with propeller installed, the pulse widths of each motor as well as the PID controller outputs , which are determined by the gyro inputs, were monitored via serial. Each axis was done independently, tilting the drone in the pitch, roll, and yaw axes, to see if the pulse widths of the motors would change as desired. Another testing method used was to fix the drone in one plane, vary the transmitter input and watch the PID controller output increase or decrease trying to reduce the error and reach its setpoint. Again, this was done for the roll, pitch and yaw axes independently. After significant tuning to the controller and pulse width equations, the drone responded as intended using the gyro to reach its angular rate setpoints.

After completing all critical milestones, it was time to begin full system tests. The full systems tests focus on achieving stabilization in each axis before testing untethered flight. Stabilization consist of prompt response of the drone to match the angular rate setpoint determined by the transmitter input. Slow response or oscillatory behavior are both signs that stabilization has not been achieved.

|  |  |
| --- | --- |
| **Full System Tests** | **Setup** |
| Drone stabilization in yaw axis | Drone hung from tethered string harness |
| Drone stabilization in roll axis | Drone fixed to freely spinning axle along roll axis |
| Drone stabilization in pitch axis | Drone fixed to freely spinning axle along pitch axis |
| Drone stabilization in all axes | Untethered drone in free flight |



The first test was done in the yaw axis, by hanging the drone from a string harness mounted to the ceiling. First the transmitter yaw input was left untouched to see if the drone would not rotate in the yaw axis at all. Then the yaw input was set to the right and left to confirm that the drone would rotate to the right or left for as long as the input was not zero. Unfortunately, when the yaw input was zero, the drone would still rotate in the counter clockwise direction. When the yaw was offset, the drone would spin in the correct direction. After offsetting the yaw and returning it to zero, the drone would stop rotating but then slowly rotate in the counter clockwise direction again. This yaw bias was eventually corrected by calibrating the middle point. Eventually, the drone successfully responded to all yaw inputs.



The second test was performed in the roll axis. The drone was mounted to an axle that could freely rotate on two bearings parallel to its roll axis. This setup confined movement to only the roll axis. Again, the transmitter input was set to zero initially to see if the drone would not rotate. Then, the roll input was set to the right or left to see if the drone would rotate to the right or left along the roll axis. Finally, the roll input was returned to zero after an offset, to confirm the drone would stop rotating and remain still. All these tests were successful, although there was a slight roll bias causing the drone to rotate slowly when the roll input was zero.

The third test was performed in the pitch axis. The drone was mounted using the same setup as the roll axis test but this time with the axle parallel to its pitch axis. The pitch input was changed the exact same ways as the roll input. These tests were also successful, but again there existed a slight pitch bias.

At this time, we are still completing tuning of each axis. Once each axis is independently stabilized, we will remove the drone from all tethered apparatus and attempt an untethered flight. The test criteria will be simple: does the drone respond to all inputs correctly, and does it fly? Even though all axes may be stabilized as shown in previous tests, the drone may not necessarily fly if the gains of the PID controller are not tuned accurately. Our untethered test will be over grass, or another soft surface and at low altitudes. We will tune the gains of the controller as we go to achieve quick reaction times and stable flight.

5 Results and Challenges

In our project proposal we aimed to use the FRDM K64 as a flight controller for a drone. We have successfully implemented a flight controller using a gyroscope. We have successfully achieved stabilization in each axis while tethered. Although we have not tested untethered flight yet, we are confident with more tuning and a few more tests we will achieve stabilized untethered flight.

In our proposal we mentioned that if we successfully completed our flight controller, we would use time-of-flight LAYDAR sensors to implement crash avoidance in certain axes. Unfortunately, we did not have time to implement crash avoidance.

Throughout the process of completing our critical milestones we ran into many challenges. At one point, the conductive carbon fiber frame shorted all the electronics on the drone. We had to disassemble the entire drone and frame, insulate all exposed wire connections, and reassemble the drone. In addition, when first powering the FRDM K64F using an external 5V regulator and the battery, we were not able to power the receiver using the FRDM’s 5V GPIO port. Finally, we found in the data sheet that when powered via the Vin port, the board does not supply 5V to the 5V port. We solved the problem by powering the receiver directly from the 5V power supply. The most complicated part of our project by far was the PID controller code. Tuning the gains, calibrations, and outputs from the controller to achieve stabilization in each axis was not easy. Having our code operate efficiently was also a huge challenge because the slower the code the worse the drone’s reaction time. The main loop had to run as quickly as possible in order to increase the resolution of the PID controller output. One solution that worked very well was removing all printf statement from our code. This greatly decreased the main loop’s runtime and increased the drone’s performance.

Our process of testing and iteration worked very well for us. The only thing we could have done differently is start a few days earlier to have been able to do an untethered flight before the report was due. We will be doing an untethered flight before the demo.

We are very proud of the work we have done, and our extremely excited to achieve untethered flight. Regardless of the final outcome, we have learned an incredible amount and have really enjoyed working on this project.