UAV Team Final Report

Rachel Bonek, Kevin Conway, Stephen Jackson, Michael O’Brien, Tom Pollei

Senior Design Spring 2019

8 May 2019

1. Table of Contents

2 Introduction

Our team has designed the unmanned aerial vehicle (UAV) payload electrical systems for Notre Dame's NASA Student Launch team. Included responsibilities are management of power, flight controller, onboard microcomputer and camera, and electronic speed controllers. Our UAV has the capability to navigate to a target where a navigational beacon shall be deployed.

The official competition description per NASA's website is as follows:

> "The NASA Student Launch (SL) is a research-based, competitive, and experiential learning project that provides relevant and cost-effective research and development."
>
> "Student Launch reaches a broad audience of colleges, universities, and secondary institutions across the nation in an 8-month commitment to design, build, launch, and fly a payload(s) and vehicle components that support NASA research on high-power rockets. The College/University Division challenge is based on team selection of one of the following experiment options: 1) deployable rover and soil sample collection or 2) deployable unmanned aerial vehicle and delivery of a navigational beacon."

This year Notre Dame Rocketry Team selected experiment option 2, a deployable UAV. NASA's Student Launch competition runs from August to April, with Launch Day scheduled for April 6, 2019. During these 8 months, the team has completed a series of design reviews that resemble the NASA engineering design lifecycle.

Ultimately, we believe our UAV design met almost all of the Notre Dame Rocket Team's and our own expectations. Our first constraints, given to us by the vehicles team, were a mass allocation of 35 ounces and that our drone must fit within a payload bay that was 7.7 inches in diameter and 20 inches long. In the final iteration our UAV was 34.7 ounces, 0.3 ounces beneath our requirement, and was able to fit successfully into the payload bay thanks to spring-loaded folding arms. The next design consideration for our team was maximizing flight time, as the beacon delivery targets could be within a circle with a one-mile diameter. By our calculations, our flight time was approximately 13 minutes, and we never ran out of battery during any of our drone test flights.

Despite initially encountering considerable challenges with binding our UAV's flight controller to a transmitter, we eventually overcame this roadblock. In the final design we were not only able to control the flight of the drone with our transmitter, but also could move a servo motor to deploy our beacons, implemented a kill switch in case of flight emergencies, and added a toggle to alternate between different flight modes. Although our drone handled fairly well during manual flight, we had hoped that the controls would have been smoother and more precise than they were. A definite cause for this unexpected result was never established, but we believe it may be due to the weight distribution of components or the form factor of the UAV itself. No issues arose in which the connection between the flight controller and transmitter was broken during flight.

One of the final practical requirements for our UAV payload was that it must be securely retained throughout the duration of the rocket's flight. To accomplish this, holes were drilled through the legs of the drone and aligned with holes in aluminum footings that attached to the deployment sled in the payload bay. Cotter pins were inserted into these holes, and as the sled moved linearly towards the nose cone during the deployment sequence these clips would slide out, allowing the drone to take off. This retention mechanism had surprising success, for during our rocket's final test flight the main parachute never deployed, and the vehicle crashed to the ground at approximately 80 ft/sec. Despite the rocket body itself sustaining heavy damage, the UAV was completed unharmed except for one broken arm. This weak point was identified, and the problem was addressed by increasing the 3D print infill from 50% to 100%.

Capturing a continuous first-person view video from our drone was a critical functionality that our team needed in order to successfully fly the UAV. For our intended autonomous flight, this video would be necessary to capture images that the Raspberry Pi could analyze and compute coordinates that could be sent to the flight controller. In manual flight, streaming this feed to a laptop at the ground station would give us a better perspective that our team could use for more accurate navigation. This task provided surprising challenges, as we had trouble finding a compatible transmitter and receiver pair for the job. Even after this hardware was acquired, we found there were extreme levels of noise in the signal once the motors started spinning, and this completely obscured the streamed video. By the end of the semester we were able to solve this issue, and our drone could successfully stream first person video even while it was in flight.

One intended design that our team was never able to implement or test was autonomous flight. Initially the team had hoped to utilize a search pattern complemented with target detection to automatically find and navigate to a beacon deployment zone. An onboard Raspberry Pi would perform all of the computations to guide the UAV to the deployment zone by performing calculations and sending corresponding coordinates to the flight controller. A computer science team was leading the effort to develop the required software, but unfortunately their progress was delayed by several bugs that took time to resolve. Once they believed all of these issues had been worked out, we attempted to integrate this functionality with our Raspberry Pi. This task introduced new complications that we were unable to sort out before our rocket's test flight during which the UAV was damaged. At this point we could not complete functional testing, as we had to wait for a new drone body to be fabricated. In hindsight had our team coordinated with the software team earlier in the design process, we may have recognized the integration issues and allocated ample time to solve them prior to our final test flight.

As the official Student Launch event was drawing close, NASA made a critical clarification, stating that on launch day teams would be allowed to scout the field and gather the GPS coordinates of the multiple beacon deployment zones. This revelation led the UAV team to develop a new and simpler software that would calculate the nearest target to the landing coordinates of our rocket. The drone would then fly directly to this location, at which point we would switch to manual flight to land the drone and deploy the beacon. However, manual flight was also allowed during competition, and NASA stated no points would be deducted for lack of autonomous flight. Thus, we focused our efforts on the more pressing task of completing the deployment mechanism.

Initially our senior design team was not responsible for developing the deployment system for the UAV payload. The underclassmen working on this component struggled greatly while attempting to get the electronics to work properly. Shortly before our final test launch, this responsibility was transitioned to our team. In the final design, a signal was transmitted from a ground station laptop that would activate the deployment process. The payload bay would be rotated by a servo motor until the on-board accelerometer confirmed the drone was facing upright, at which point a gear motor would begin to push the nose cone off of the rocket and UAV sled out of the rocket. These systems worked as anticipated, but damage to several components prevented our team from testing the deployment mechanism post flight. Right before our rocket test flight, the wires from the orientation servo got tangled and torn. Furthermore, the rear bulkhead was geared to enable orientation from the servo, and some of this gearing was faulty and would cause the servo to get stuck.

In summary, our final project worked mostly as anticipated, with several minor exceptions. First, the drone controls weren't as refined as we had hoped, and this was likely due to the physical composition of the UAV. Second, the team was never able to integrate automatic target detection and navigation as we originally intended, although this was not necessary for our payload to be successful in competition. Lastly, we were never able to test our payload deployment and flight after a test launch. Portions were damaged during our final rocket test launch, and the resulting damage prevented us from completing another launch once the competition had ended. Although our team had established a majority of the subsystems functioned independently as intended, we never had the opportunity to ensure they all worked when they were incorporated in a full test launch.

3 Detailed System Requirements

Requirements for the deployable UAV taking from the Student Launch Instruction handbook are as follows:

- 4.4.1. Teams will design a custom UAV that will deploy from the internal structure of the launch vehicle.

- 4.4.2. The UAV will be powered off until the rocket has safely landed on the ground and is capable of being powered on remotely after landing.

- 4.4.3. The UAV will be retained within the vehicle utilizing a fail-safe active retention
system. The retention system will be robust enough to retain the UAV if atypical flight
forces are experienced.

- 4.4.4. At landing, and under the supervision of the Remote Deployment Officer, the team will remotely activate a trigger to deploy the UAV from the rocket.

- 4.4.5. After deployment and from a position on the ground, the UAV will take off and fly to a NASA specified location, called the Future Excursion Area (FEA). Both autonomous and piloted flight are permissible but all reorientation or unpacking maneuvers must be autonomous.

- 4.4.6. The FEA will be approximately 10 ft. x 10 ft. and constructed of a color which stands out against the ground.

- 4.4.7. One or more FEA's will be located in the recovery area of the launch field. FEA samples will be provided to teams upon acceptance and prior to PDR.

- 4.4.8. Once the UAV has reached the FEA, it will place or drop a simulated navigational beacon on the target area.

- 4.4.9. The simulated navigational beacon will be designed and built by each team and will be a minimum of 1 in W x 1 in H x 1 in D. The school name must be located on the external surface of the beacon.

- 4.4.10. Teams will ensure the UAV's batteries are sufficiently protected from impact with the ground.

- 4.4.11. The batteries powering the UAV will be brightly colored, clearly marked as a fire hazard, and easily distinguishable from other UAV parts.

- 4.4.12. The team will abide by all applicable FAA regulations, including the FAA's Special Rule for Model Aircraft (Public Law 112-95 Section 336; see https://www.faa.gov/uas/faqs).

- 4.4.13. Any UAV weighing more than .55 lbs. will be registered with the FAA and the registration number marked on the vehicle.

4 Detailed project description

For sections 4.1 and 4.2, please refer to the relevant sections in the Notre Dame Rocket Team Flight Readiness Review document located on the website, specifically pp. 64-77 and 85-92, for more information.

4.1 System theory of operation

4.2 System Block diagram

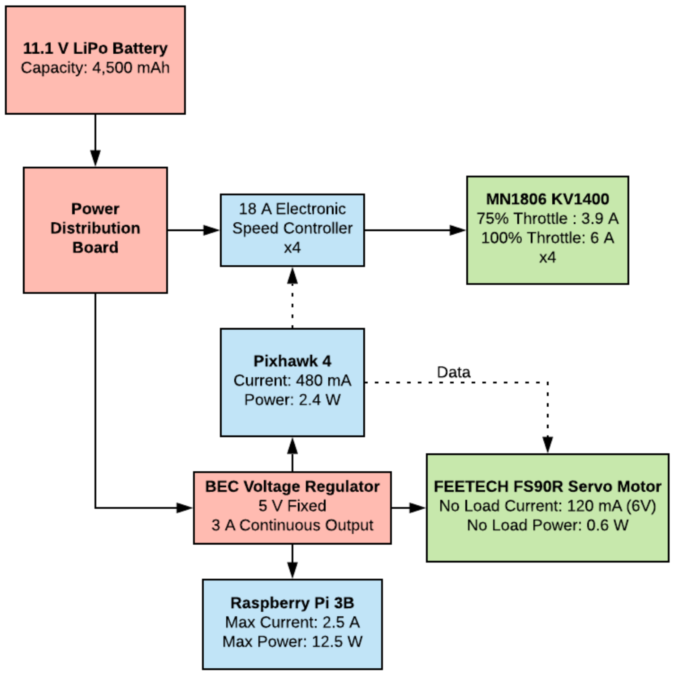


Figure 1. System Block Diagram.

4.3 *Subsystem 1: Deployable Drone Communication System*

4.3.1 Communication System Overview

The requirements for the communication systems are as follows:

- ❖ 4.4.5. After deployment and from a position on the ground, the UAV will take off and fly to a NASA specified location, called the Future Excursion Area (FEA). Both autonomous and piloted flight are permissible but all reorientation or unpacking maneuvers must be autonomous.
- ❖ 4.4.6. The FEA will be approximately 10 ft. x 10 ft. and constructed of a color which stands out against the ground.
- ❖ 4.4.7. One or more FEA's will be located in the recovery area of the launch field. FEA samples will be provided to teams upon acceptance and prior to PDR.
- ❖ 4.4.12. The team will abide by all applicable FAA regulations, including the FAA's Special Rule for Model Aircraft (Public Law 112-95 Section 336; see https://www.faa.gov/uas/faqs).

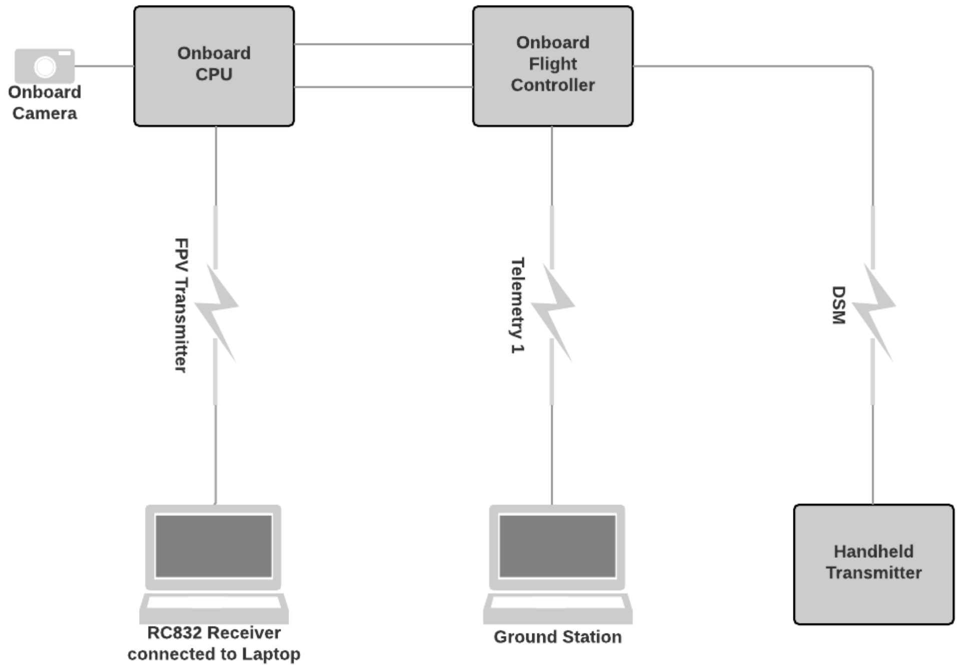


Figure 2. Communication Architecture

The UAV system will be designed such that it will be able to fulfill its mission with complete autonomy. However, the system will also have a redundancy such that a switchover to manual flight control is possible. The UAV will fly a preprogrammed flight plan upon deployment from the drone. During flight, the onboard CPU, a Raspberry Pi 3 Model B, will process data from the onboard camera using a search algorithm to detect the target. Once the target has been detected, the onboard CPU will upload a new flight plan to the flight controller, a Pixhawk 4.

In the interests of redundancy, the onboard CPU will stream the visual data via telemetry from the onboard camera to a CPU on the ground (also a Raspberry Pi 3 Model B) which will display the data for first person view. There will also be a telemetry link between the flight controller and the ground station, a laptop. This will provide real time spatial coordinates of the UAV visible on Google Maps. Lastly, there will be a handheld controller for use in the case where manual takeover is deemed necessary.

An overview of the Communication System architecture is visible in Figure 2.

4.3.2 Selected Components

Central Processing Unit (CPU): Raspberry Pi 3 Model B (Raspberry Pi 3B)
Since several team members have utilized Raspberry Pi's in previous projects, the team chose this platform for the UAV's onboard CPU. The following are two models the team primarily researched:

a. Raspberry Pi 0
    i. Pi 0 streams information to the ground station where the target detection algorithm is processed; subsequently these coordinates are transmitted back to the Pi 0 and sent to the flight controller.
b. Raspberry Pi 3B
    i. Onboard Pi 3B performs all target detection processing and sends coordinates to the flight controller. This solution protects from communication loss with the ground station. The team selected this CPU for the UAV due to the reduced complexity in wireless communication.

Flight Controller: Pixhawk 4
The team selected the Pixhawk line of flight controllers based on brand reputation and the prior experience of team members. Within this line, the following models were considered:

c. Pixhawk Mini

   i. The team considered utilizing the Pixhawk Mini as a small and lightweight solution, but this model was discontinued by the manufacturer.

d. Pixhawk Falcon
   i. The Pixhawk Falcon was rebranding of Pixhawk mini, providing a lightweight flight controller with a small form factor and low power consumption. One of the main drawbacks of this option is its lack of connection ports.

e. Pixhawk 4
   i. The Pixhawk 4 is faster, has onboard heating for cold weather testing, and has more connection ports, including a servo rail that is useful for power splitting. Despite having several more features, this option is not significantly larger or heavier than the Pixhawk Falcon. Thus, the team selected this model to use for the UAV.

Ground station

For the ground station, the team will be utilizing two separate laptops for increased viewing screen real estate. One will be receiving video stream from telemetry with the Raspberry Pi 3B, and the other will be utilized to observe real time UAV coordinates and control inputs during optional autonomous flight.

Handheld transmitter

The main metrics for the transmitter were compatibility with the Pixhawk and the cost. The Spektrum Dx6e was initially chosen because it is more economical compared to other Spektrum transmitters. Also considered was the Taranis Qx7, because of its price point. However, the transmitter does not transmit using the Digital Spectrum Modulation (DSM/DSMX) communication protocol and would require additional adapters and converters that add cost and weight to the UAV. In the end, the FrSky Taranis X9D Plus 2.4GHz Advanced Continuous Channel Shifting Technology (ACCST) Radio was chosen because the team's faculty specialist in drones donated the handheld transmitter for the team's use. The Taranis has full telemetry capability, three separate programmable fail-safe modes, and a JR style Radio Frequency (RF) module slot. The transmitter communicated with the UAV through a FrSky (pronounced Free Sky) X8R receiver that included two antennas and access to all 8 channels through a pin-out.

Camera: Pi Camera Module V2

As the Raspberry Pi 3B was selected as the CPU for the UAV, the team decided to primarily research the cameras designed specifically for this device. Options included:

a. Pi Camera Module V2

    i. This model is standard camera module that integrated with Raspberry Pi CPUs. It provides 8 megapixels of resolution and can take high resolution videos as well as still photographs. The team would use this device to stream constant video to the Raspberry Pi 3B, which can then be used either for target detection or manual flight. Thus, the team selected this model.

b. Pi Noir Camera V2
    i. This camera is identical to the Pi Camera Module V2, except it lacks an infrared filter, which provides the ability to see in the dark. However, the lack of this filter makes vision in daylight somewhat more difficult. Since the UAV would ideally be operating in the brighter conditions of Alabama, this feature would be detrimental to overall targeting functionality.

FPV Transmitter: Eachine VTX03 Super Mini 5.8GHz 72CH
The transmitter for the first-person view system on the drone is the Eachine VTX03 Super Mini 5.8GHz 72CH FPV Transmitter. We selected this over the 915MHz telemetry set because the later would have resulted in a video stream with extremely high latency. The Eachine transmitter has three output power settings: 25mW, 50mW, and 200mW which makes it ideal for testing. A low power setting can be used for close range testing, and the high-power setting, i.e. 200mW, can be used for actual flight during competition.

FPV Transmitter: RC832 5.8G AV Receiver
This receiver can be directly connected to a laptop via a USB adapter. Both the transmitter and the receiver have multiple selectable channels in the 5.8GHz band. The channel that has been chosen is B7, or 5.847GHz.

4.3.3 Testing

We tested the FPV connection by first connecting a monitor directly to the Raspberry Pi and viewing video via an application, we ensured that the camera module was successfully capturing footage. Subsequently, viewing this footage on a laptop through telemetry confirmed that we were able to stream video wirelessly from the UAV. Finally, by testing this functionality with the spinning motors, we established our ability to perform this task during flight, which was our final goal.

To test our camera streaming, we first installed the Raspberry Pi camera module onto the parent device. We then connected a monitor to the Raspberry Pi and ran a program, opening an application that displayed the video feed captured by the camera with some latency. The next step was to send this feed wirelessly via telemetry from the drone to the ground station. We were able to move the camera around and see the video on the laptop display was updating. Next, we spun up the motors and attempted this process again but noticed that this created noise that completely obscured the video. With adjustments, we were able to view streamed video on the laptop with minimal noise even when the motors were spinning and the drone was in flight.

To pair the flight controller to a ground station laptop, telemetry had to be established on both ends of communication. This requirement was achieved through the use of matched antennas. In order to ensure successful connection, we had first had to find our flight controller on the ground station software, QGroundControl. Once this was accomplished, the flight controller made a sound, indicating the pairing was made. We then uploaded a command to calibrate the flight controller and GPS module, completing the subsequent steps by repositioning and rotating the UAV. The QGroundControl acknowledged this calibration and notified us that the process was successful.

To confirm communication between the Raspberry Pi and the Pixhawk 4, we borrowed a basic program from a graduate student who specializes in working with drones. This drone read parameters directly from the flight controller. We connected a monitor to the Raspberry Pi and uploaded this code. Upon running it, we were able to observe orientation parameters. We would reposition the drone, run the program again, and see that the previous orientation values had changed.

4.4 *Subsystem 2: Deployable Drone Power Management*

4.4.1 Requirements

• 4.4.1. Teams will design a custom UAV that will deploy from the internal structure of the launch vehicle.

• 4.4.2. The UAV will be powered off until the rocket has safely landed on the ground and is capable of being powered on remotely after landing.

• 4.4.3. The UAV will be retained within the vehicle utilizing a fail-safe active retention
system. The retention system will be robust enough to retain the UAV if atypical flight
forces are experienced.

• 4.4.4. At landing, and under the supervision of the Remote Deployment Officer, the team will remotely activate a trigger to deploy the UAV from the rocket.

• 4.4.5. After deployment and from a position on the ground, the UAV will take off and fly to a NASA specified location, called the Future Excursion Area (FEA). Both autonomous and piloted flight are permissible but all reorientation or unpacking maneuvers must be autonomous.

• 4.4.10. Teams will ensure the UAV's batteries are sufficiently protected from impact
with the ground.

• 4.4.11. The batteries powering the UAV will be brightly colored, clearly marked as a
fire hazard, and easily distinguishable from other UAV parts.

• 4.4.13. Any UAV weighing more than .55 lbs. will be registered with the FAA and the registration number marked on the vehicle.

4.4.2 Design Iteration Process

For the selection of power components an iterative design process was used. The iteration started with the motor, followed by the props, the electronic speed controllers (ESCs), and the lithium polymer (LiPo) battery, as shown in Figure 3. The following design explanations and decisions pertain to requirement 4.4.1.

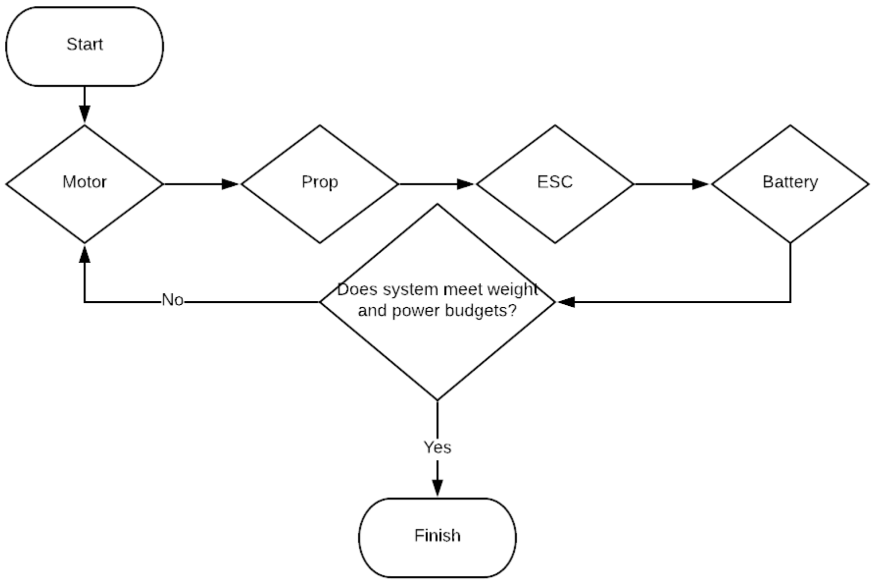


Figure 3. Design Process Flow Diagram

1. Find a system of 4 motors with 30-40oz of thrust at 70% thrust (from T Motor site).
In order to fly, the motors need to be capable of generating thrust equal to the drone’s weight. However, the drone will not be able to get off the ground if the maximum thrust available is equal to weight, so the motors need a little bit of clearance to increase altitude. Therefore, a system of motors capable of generating 30-40 oz. of thrust at 70% throttle was specified. Choosing the 70% point as opposed to a 50% or even 30% power point would provide smoother controls during flight. Full throttle is rarely utilized except for maximum ground speed. Instead, throttle is centered around the hover point and typically stays within 15% of hover. By choosing 70% throttle as the hover point, the throttle will prove to be much less sensitive.

2. Choose prop size based on motor specifications
Once a motor of sufficient thrust with some given values of current and voltage was selected, the next step was to select props. Motor specifications provide information on performance (thrust and current draw) with respect to different propellers sizes. That information was used to determine the length and pitch of the props. The size constraints provided by the length of the payload bay of 20” and payload bay diameter of 7.7” were used as the final constraint to guide prop selection.

3. Select an ESC
With motor and prop dimensions in hand, the power requirement and current draw of the motors was determined. This dictated the rating required for the ESC. The ESC is essentially a smart fuse. It must be capable of withstanding the current draw from the motor, but it must not be so big that it adds unnecessary weight to the system.

4. Select a battery based on voltage requirement
Finally, once the power components were chosen, the battery could be selected. Considerations for the battery were voltage necessary to drive motors, capacity for maximizing flight time, C rating for maximum current draw, and overall physical weight and size. When determining the max current draw, onboard controller needs were estimated and to the total current draw.

Once a system was chosen, its performance was considered against the constraints imposed by the requirements of other systems. If some aspect of the system exceeded limits or failed to meet the specifications, the process was repeated to correct that factor. Many iterations were necessary. For example, a larger battery was chosen to generate longer flight time, however the battery account for an estimated 50% of the total weight of the system. In addition, shorter props were chosen to better fit inside the rocket.

4.4.3 Selected Components

Motors

The three motors considered were all products from T-Motor, a trusted and reputable Radio Controlled (RC) motor company. These were the MN1806, MN2212, and Antigravity 4004. First-Person View (racing drones) motors were also looked at because they offer a very high thrust to weight ratio, however the current draw is too high for the system and would require a battery that would have been outside the weight allowance. The Antigravity motor was considered because it provides the most thrust per Watt, however it was not chosen because the total thrust provided was more than what was needed, even on the lowest end. Comparing the MN1806 to the MN2212, the MN1806 fit the 30-40 oz thrust range, and it also outperformed the MN2212 in thrust per watt while weighing 45g less per motor. For this reason, the team moved forward with the MN1806 KV1400 motor from T-Motor.

Props

The options for props were narrowed down greatly by the specifications for the selected motor. The motor chosen provides data for props between 5 and 9 inches. The prop ultimately selected was the Multirotor Carbon Fiber T-style Propeller 7x2.4, a 7-inch prop with a 24-degree pitch. This was chosen because of its length, thrust capability, and required battery size. It is a two-blade prop that is short enough to fit into the payload bay when implemented into the body design. Also considered were longer props and four-blade props. Their advantage was greater thrust at a given voltage. However, they were too large to fit in the payload bay and had to be discounted. When comparing the 7-inch option to the smaller size options, the 7-inch prop was the only prop size that provided enough thrust.

Electronic Speed Controllers

The ESC ultimately selected was the Lumenier 18A 32bit Silk ESC opto-isolator (OPTO) (2-4s). This was selected primarily based on its amperage rating, corresponding to the required amperage rating of the selected motor. Each motor draws 6A at 100% throttle but can draw up to 12A maximum continuous current during peak moments such as takeoff and direction changes. 18A was large enough to withstand 12A burst currents, but not too large that it added unneeded weight to the system.

Battery

The battery ultimately selected was the Turnigy Nano-tech 4500mAh 3S Lithium Polymer battery. This provided enough power for the selected ESC's and motors for the desired flight time (around 9-14 minutes). 4,500 mAh was an upgrade in terms of capacity from the original 3,000 mAh battery once it became clear that longer flight time was needed.

Requirement 4.4.2 required that "the UAV will be powered off until the rocket has safely landed on the ground and is capable of being powered on remotely after landing." Thus, the team decided to use a toggle switch to control the power-on sequence. The switch will isolate the battery while open and will control current flow from the battery to the rest of the drone. The switch will be in the "off" position when it is inserted into the rocket. A cord fixed to the bulkhead will then be attached to the lever of the toggle switch. As the drone is deployed from the rocket, this switch will be activated and pulled into the "on" position. The cord will then disconnect from the lever, leaving the drone ready for takeoff.

4.4.4 Testing

The first subsystem that had to be implemented was the power distribution board (PDB). This system was a necessary first step, as without it power from the battery could not be properly allocated to the other subsystems: the flight controller, the GPS module, electronic speed controllers, motors, Raspberry Pi, and beacon deployment servo motor. To test the PDB, we initially read the voltages on the various pins and pads with a voltmeter. These observed values were then compared with those provided in the specification sheets to ensure they matched. Once values were confirmed, we soldered the appropriate wires and made the necessary connections. Our team then observed the behavior of the flight controller, GPS module, and electronic speed controllers (ESCs). The expected lights were glowing on all components, giving an initial indication that they were connected correctly. The ESCs began beeping, however, and we noticed that they were missing a PWM signal connection. We resolved this issue, and the error stopped.

By first checking the continuity of the terminals of the toggle switch, we confirmed that the device operated as was expected. Once connected between the LiPo battery and PDB, we saw that power never reached the PDB when the switch was in the off position. This displayed the desired functionality that would allow us to keep the drone in an unpowered state during flight. Once switched to the on position, all subsystems were activated, showing initial signs they were functioning as intended. Reading the voltage on both ends of the switch at this point let us observe that the power supplied to the PDB was unaffected. Lastly, by flying the drone with the switch integrated, we were ensured that it was able to withstand the heavy current drawn by the motors.

5 System Integration Testing

5.1 Describe how the integrated set of subsystems was tested.

5.1.1 Pairing with Transmitter

In order to fly the drone, a manual control transmitter must be bound with the flight controller. To test this, we first observed the behavior of the GPS module, which in the specifications document said would light up a specific color when successfully paired with a transmitter. Once a compatible transmitter and receiver for the UAV were obtained, we held a button on the GPS module and the transmitter to begin the synchronization process. After a few seconds, both devices emitted notification sounds, indicating the binding was successful. We then armed the UAV, and the motors started spinning. When the controls rods were moved on the transmitter, the speeds of the separate motors would noticeably change.

5.1.2 Functionality of Electronic Speed Controllers (ESCs)

Although we were able to see that the ESCs were receiving power from the PDB, we were unsure if they could properly interpret PWM signals from the flight controller to regulate the speeds of the four motors. Motors diagonal from one another are supposed to rotate in the same orientation as one another, and two must rotate clockwise while the other two rotate counterclockwise. Once our transmitter was bound with the UAV, their functionality and orientation could be tested. We attached flaps of tape to each of the four motor mounts and armed the drone with the transmitter, causing them to spin up. All motors spun but one, so we replaced the corresponding ESC and this issue was resolved. The motors were not rotating in the desired orientations, however, so by changing their connections to the ESCs the proper orientations were obtained. Once the these were corrected, we armed the drone again and began moving the control rods on the transmitter. Subsequently, the speeds of the various motors would change. We then attached props to the motors and repeated the process, and the drone would move in the directions that we dictated via the transmitter.

5.1.3 Integrating Power Regulating Toggle Switch

Since NASA required that our drone could be powered down during rocket flight, we chose to add this functionality with a toggle switch. To test, we first used a voltmeter to read continuity on the terminals of the switch. When in the off position, no continuity was observed between the paired terminals. Switched to the on position, this continuity was confirmed. Next, we connected the switch between the battery and power distribution board. The switch was activated, and the voltage was read on the end attached to the battery and the end attached to the PDB. These readings were identical. Simultaneously, we saw that the various subsystems began to power on as normal, indicating that they were receiving the power needed to operate. The drone was then flown with the switch implemented, and it performed as expected.

5.1.4 Integrating Beacon Deployment Servo

The final subsystem that was integrated into our drone was the beacon deployment servo. This was connected directly to the FrSky X8R receiver, and signals from the Taranis X9D transmitter control the movement of the motor. A dial was programmed to communicate over channel 5 to the receiver, and the servo was expected to connect to channel 5 on the receiver. Figure 4 below shows the FrSky X8R receiver. The motor required 5V, so we used a voltmeter to read the corresponding pin and confirm this voltage level. The servo responds to PWM signals from the signal pin. A voltmeter was connected to the signal pin as the dial was adjusted. The voltmeter responded accordingly to the dial inputs. The servo was then connected to channel 5 and its response to dial inputs was noted.

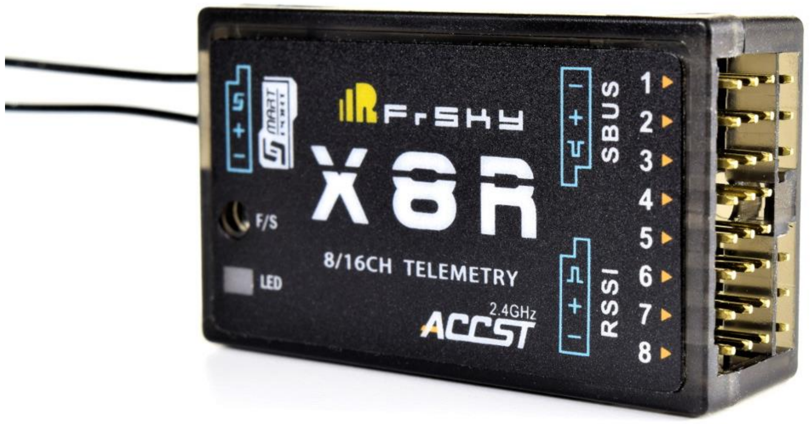


Figure 4. FrSky X8R Receiver. The external pins for all 8 channels are easily accessible.

5.2 *Show how the testing demonstrates that the overall system meets the design requirements*

5.2.1 Power Distribution Board

By reading and confirming the voltages on the terminals of the PDB, we knew that the battery was successfully supplying power to the board and that it was being properly allocated. Observing the initial behavior of the various subsystems connected to the PDB notified us that these components were receiving the necessary power for operation.

5.2.2 Pairing Flight Controller with Ground Station

When we found the flight controller on QGroundControl and connected to it, the sound from the Pixhawk indicated that an initial communication link had been opened. By uploading a calibration command to the UAV from the ground station, we confirmed that signals could successfully be sent from a laptop and read by the flight controller. When QGroundControl acknowledged the various calibration procedures of the drone, we confirmed that signals could be sent from the drone and interpreted by the ground station.

5.2.3 Pairing with Transmitter

When the transmitter and GPS module emitted the desired colors and produced notification sounds, we knew that the drone and transmitter had been successfully paired. By arming the motors with the transmitter and causing them to spin, it was confirmed that the flight controller was receiving commands properly from the transmitter. Furthermore, by moving the control rods on the transmitter and observing speed changes in the motors, it was determined we could successfully control them for flight.

5.2.4 Functionality of Electronic Speed Controllers (ESCs)

Upon arming the drone and causing the motors to spin, we could see that all ESCs but one were able to successfully interpret PWM signals from the flight controller and regulate the four motors. It was determined this malfunctioning ESC was damaged during soldering and replaced to resolve the issue. With the tape, we could observe the orientations of our motors and correct them so they would spin as intended which is necessary for flight. Next, by noting motor speed changes when the control rods were moved, we determined that the ESCs could receive signals from the transmitter and control motor speeds to display the desired characteristics for flight. Lastly, with the props attached, we ensured that our manual controls were functioning properly and suitable for flight.

5.2.5 Communication between Raspberry Pi and Flight Controller

By utilizing this code, we were successfully able to read parameters from the flight controller with the Raspberry Pi. This functionality confirmed that there was a successful communication link between the two components, and that we were able to upload commands to the Pixhawk. Furthermore, the resulting signals could be transmitted back the Raspberry Pi.

5.2.6 Integrating Beacon Deployment Servo

By reading the voltage level of the servo rail pin, we were able to confirm that it had the necessary 5V supply to operate. Moving the dial on our transmitter and seeing the movement of the servo established that the corresponding communication channel operated, that the flight controller was able to send PWM signals to the servo, and that the motor itself could interpret these signals and move in the desired directions. Upon connection, it was discovered that the channel had an offset of 0.1 V which would cause the servo to rotate when in the off position. To solve this problem, settings were edited from the transmitter side. The transmitter allows curves and offsets to be defined for translation between dial position and output value. The translation curve was shifted down to center the off position. In addition, the motor spun too fast and was hard to control by turning a dial. To counter this, the slope of the output curve was decreased. Now, the system is verified when the servo is motionless when the dial is at center and when it slowly responds to dial inputs.

6 User’s Manual

6.1 *How to setup your product*

6.1.1 Binding Drone Transmitter and Receiver

With the receiver <u>off</u>, power on the transmitter, press the Menu button, and toggle into Model Setup using the Page button. Use the - button to toggle down to Internal RF. Mode should be D16, Channel Range should be 1-8, Receiver No. should be 2. Keep pressing - until Bind is highlighted, then press the Ent button. Now, with the receiver still off, short out the top wires (i.e. the signal pins) of Channels 3 and 4 (a small black jumper piece is provided for this purpose). Then, press the F/S button on the receiver, and then connect it to power (e.g. plug in the Pixhawk) while still holding the button down. Hold for a few seconds. To ensure that pairing has actually occurred, turn off both the transmitter and receiver, then turn both back on. Once the transmitter is fully on (i.e. after its startup procedure), the LED on the receiver should be a solid green. Turning off the transmitter should make the receiver's LED flash red. Failsafe mode can be enabled after the binding process is complete. "Failsafe is a useful feature in which all controls revert to a preset position in the case of a lost control signal" (from the receiver manual). To enable failsafe, first move the controls on the transmitter to what you want the UAV to revert to in the case of a lost signal. Then *briefly* press the F/S button once. The green LED should blink twice. Failsafe mode is now enabled. To reset the failsafe presets, the binding procedure must be redone as indicated above, and then enable failsafe with the different desired settings on the transmitter.

6.1.2 FPV System

The 5.8 GHz NTSC transmitter needs to be powered from the Pi. The red (power) wire can be attached to either of the +5V pins, and the ground wire can be attached to any ground pin. The 3.5mm jack is plugged in to the corresponding port on the Pi. (For competition the 3.5mm plug will be replaced by a direct solder to the appropriate pins on the bottom of the board.) See Figure 5 below.

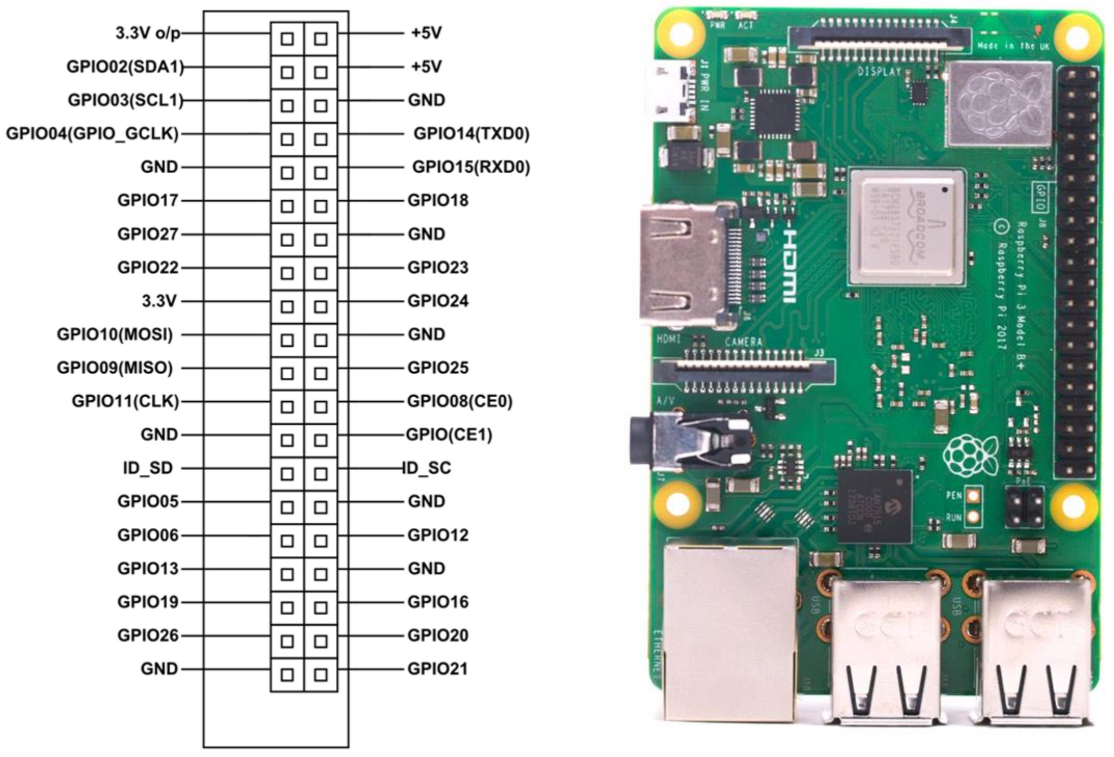


Figure 5: Raspberry Pi3B pinout schematic

The channel on the FPV transmitter has already been set to “B7” on both the transmitter and receiver, so there should be no need to change it. (“B7” corresponds to 5.874 GHz). To change the transmit power, press the button on the transmitter for 5 seconds, then short press the button to toggle through the power settings. ‘0’ corresponds to no power, one bar to 25mW, two bars to 50mW, and three bars to 200mW. Note: pressing the button with short presses will change the channel selection from 1 to 8. Pressing the button for 2 seconds will give you the option to toggle through the frequency band selection (A, b, E, F, etc.)

To view the video stream on a laptop, connect the FPV receiver via the USB adaptor. There is a second USB plug for power only. The first USB adaptor comes with a special software available on CD-ROM. Install the software, plug in the USB and follow the instructions that come with the UCEC Video Capture software.

6.1.3 Arming and Disarming the Drone

The drone has a standard arming procedure, assuming there are not errors preventing arming.

1. Make sure drone is disconnected from battery
2. Point drone away from nearby people
3. Turn on transmitter with throttle stick at zero (left stick)
4. Plug in the battery and power on the system via the toggle switch
5. With one person holding transmitter throttle at zero, second person disengages safety switch by pressing the button on the GPS
6. Person 2 immediately distances from drone
7. Pilot places throttle stick in down-right position.
8. UAV will arm and motors will spin at an idle speed.

Standard disarming procedure:

1. Once the drone has landed, pilot places throttle in down-left position.
2. Person 2 switches off the toggle switch and unplugs the battery

5.2.4 Telemetry: Updating Firmware and Power Settings

To update the firmware on the Holybro telemetry sets, start with everything unplugged from your computer, open QGroundControl and hit the gear icon in the top left corner of the screen. Click on Firmware, and then follow the instructions on the screen. If firmware update is unsuccessful, this may be a sign of a faulty unit.

NASA requires that we do not exceed 250mW of output power while transmitting. To adjust the power output of the telemetry, use Mission Planner. Go to Initial Setup > Optional Hardware > SiK Radio. Plug in the telemetry unit you wish to adjust, and make sure that the COM port selection in the top right of the Mission Planner screen is correct, with a baud rate of 57600. Then click on Load Settings. The Tx Power is a drop-down menu. After selecting the desired setting, click on Save Settings. Note: There seems to be some discrepancy in what the settings mean. Most of the "official" 3DR radios (i.e. the ones not made in China) have a max output of 100mW. The highest value in the Tx Power drop-down menu is 20, i.e. 20 dBm, which corresponds to 100mW. The present setting on the telemetry kits is 17. If the settings map linearly from a 100mW unit to a 500mW unit, then ($10^{1.7}$= 50mW) x 5 = 250mW.

6.2 *How the user can tell if the product is working*

There are multiple capabilities that must be demonstrated correctly to characterize the UAV as working properly.

6.2.1 Stable Flight

First and foremost, the UAV must be able to fly. If all the props are spinning in the correct direction, and the drone can take off from the ground and maintain stable flight, the product is working. To test if the drone can maintain stable flight, throttle should be slowly increased until the UAV is just barely off the ground. The user should note that the UAV is not vibrating violently and does not drift or yaw in any direction without the appropriate input.

6.2.2 Video Streaming

After the UAV is powered on, the Raspberry Pi will automatically turn on the camera and attempt to establish a connection with the receiver. The user should see the Raspberry Pi startup screen, and after 30 seconds the computer should display a live feed.

6.2.3 Servo Motor

Lastly, the user should note that the servo motor responds correctly to the dial input. The motor should be stationary when the dial is centered and should rotate slowly when the dial is spun. The servo can spin either clockwise or counterclockwise according to the direction the dial is turned. Test that this is true.

6.3 How the user can troubleshoot the product

6.3.1 Troubleshooting before UAV Flight

The Pixhawk flight controller has many parameters. In order for the UAV to arm, all the parameters must meet or exceed acceptable values. Once connected to drone, click on the settings icon, then click on parameters. You can search parameters with the search bar. Listed below are various common issues.

1. Flashing Blue / No GPS Lock
   a. Search “GPS”
   b. Set **CBRK_GPSFAIL** to 240024
2. Mag Sensors Inconsistent
   a. First, attempt to recalibrate the UAV
      i. Open “sensors” on the QGroundControl panel
      ii. Follow the directions to recalibrate the Compass, Accelerometer, Gyroscope, and Level Horizon
   b. If calibration does not solve the issue, **COM_ARM_MAG** and **COM_ARM_EKF_AB** can be tweaked to allow for larger error offsets.

7 To-Market Design Changes

It should be noted first that this product was designed for a very specific application. However, if the UAV were to be brought to market, we would likely make several changes to the overall design. First and foremost, we would alter the design of the drone body itself. Currently, the frame is not particularly aerodynamic and leaves many of the components exposed. In a future product, the frame should be streamlined, using more rounded edges as opposed to the boxier design that exist now. This would enable smoother flight while also reducing the form factor of the UAV. Furthermore, the UAV could benefit from a lightweight exterior casing that would house all of the core electronics. With this addition, the components would be protected from damage that could occur from rain or other adverse weather conditions, as well as prevent any possibility of wires being cut from the spinning props.

Another design change would be a transmitter that has a built-in display that would allow the user to view the FPV streamed from the UAV. This change would eliminate the need for an extra ground station laptop that was needed exclusively to view the video capturing by the UAV. Furthermore, this display would likely make it easier for the operator to fly the drone, as they would simply look down to view this stream rather than having to switch between looking at the drone itself, the transmitter, and a laptop display. Overall, this to market alteration would reduce the overall complexity of the project.

In addition to these changes, the UAV could benefit from different antenna pairings than those currently in use. At the moment, the antennas take up considerable space and are located on multiple locations across the drone's body. A further reason for making this alteration is that some antennas protrude from the UAV and overlap with props, adding potential risk of damaging them during flight. The number of antennas could also likely be reduced, especially by using a transmitter with a display as mentioned above. This would reduce the number of components required for ground station and on the UAV itself, further simplifying the design of the product.

A third to market design revision that could benefit the UAV would be to use a different camera module for gathering video from the drone. Although the Raspberry Pi camera module was sufficient for the project's needs, it was stationary and did not have gimbal capabilities, not to mention poor image quality. With a less limited budget, the UAV could have utilized a higher resolution camera that could rotate and swivel the camera head vertically. This change would allow the user to stream clearer video and gather a more comprehensive view of the UAV's surroundings. With this wider field of view, an operator could better see what is ahead of or to the sides of the drone rather than being limited to seeing only that which is directly beneath it.
A fourth to market design revision would be to swap the Pixhawk 4 for the Pixhawk 2. The world of flight controllers can be very confusing to new consumers. The Pixhawk Project created the hardware standard for open source autopilots. The result is the FMUv*X*, an electrical schematic that is intended to be best suited for drone builds. It is open source and any group can create their own hardware from the electrical designs. PX4 is firmware developed by this same Pixhawk Project that is also open source. Pixhawk 2 is subsequently the name of a device and uses FMUv3 and PX4. It is the most common Pixhawk and has the most support online. In addition, the product is very reliable. The Pixhawk 4 (another device) uses Pixhawk Project hardware design FMUv5 as well as the PX4 software. It is not very compatible with the Mission Planner and QGroundControl software (also developed by the same group). In the future, all UAVs should be integrated with a Pixhawk 2.1 Cube and not the Pixhawk 4.

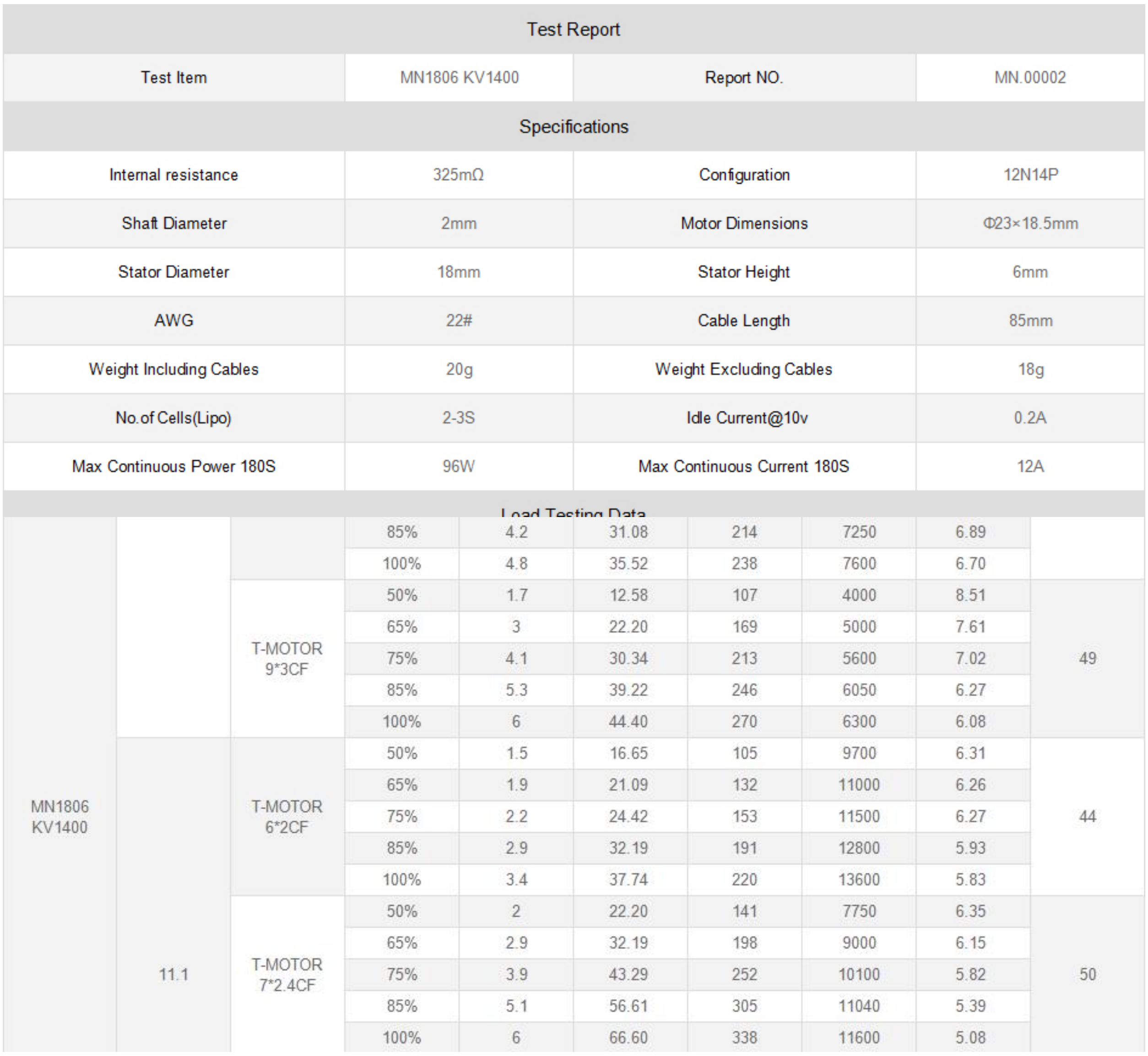
Lastly, we would have implemented a low power mode for the drone, eliminating the need for a bulky toggle switch. Since the battery provided substantial current, a rather large switch was required, and this added unnecessary weight and complexity to the design. The toggle arm itself was also rather stiff and demanded considerable force to move from the off to the on position. A low power mode would prevent battery from being consumed while not in use and could have been activated with a remote signal from the ground station or transmitter.

8 Conclusions

Despite the rocket crash on 23 March 2019 which compromised the structural integrity of the rocket, thus disqualifying the Notre Dame Rocket Team from competing in Huntsville, AL, working on this project has provided extremely valuable and irreplaceable experience to each of the team members. The very framework of the student launch was a formation of its own, from the proposals and design reviews to video presentations given to NASA representatives. Throughout the course of the year, we had the opportunity to work and interact with engineering students from other disciplines, including mechanical and aerospace engineering majors, and computer science majors. The culmination of the yearlong design process found us at the Marshall Space Flight Center in Huntsville, rubbing shoulders with students from universities and colleges throughout the United States, and learning firsthand about the great legacy of American space exploration. The most important takeaways from this senior design project are not only to be found in the low-level details, which nevertheless shaped and sharpened our technical expertise, but also in the overarching picture wherein we aspire to noble and lofty goals and reach for the stars, both literally and figuratively. *Tu dux ad astra, et semita, Sis meta nostris cordibus, Sis lacrymarum gaudium, Sis dulce vitae praemium.*

9 Appendices

1. Pixhawk 4 Flight Controller
   https://docs.px4.io/en/flight_controller/pixhawk4.html
2. Taranis X9D RC Transmitter
   https://www.frsky-rc.com/taranis-x9d-plus-2/
3. T-Motor MN1806 KV 1400



Test Report

Test Item	MN1806 KV1400	Report NO.	MN.00002

Specifications

Internal resistance	325mΩ	Configuration	12N14P
Shaft Diameter	2mm	Motor Dimensions	Φ23×18.5mm
Stator Diameter	18mm	Stator Height	6mm
AWG	22#	Cable Length	85mm
Weight Including Cables	20g	Weight Excluding Cables	18g
No.of Cells(Lipo)	2-3S	Idle Current@10v	0.2A
Max Continuous Power 180S	96W	Max Continuous Current 180S	12A

Load Testing Data

MN1806 KV1400			85%	4.2	31.08	214	7250	6.89	
			100%	4.8	35.52	238	7600	6.70	
		T-MOTOR 9*3CF	50%	1.7	12.58	107	4000	8.51	49
			65%	3	22.20	169	5000	7.61	
			75%	4.1	30.34	213	5600	7.02	
			85%	5.3	39.22	246	6050	6.27	
			100%	6	44.40	270	6300	6.08	
	11.1	T-MOTOR 6*2CF	50%	1.5	16.65	105	9700	6.31	44
			65%	1.9	21.09	132	11000	6.26	
			75%	2.2	24.42	153	11500	6.27	
			85%	2.9	32.19	191	12800	5.93	
			100%	3.4	37.74	220	13600	5.83	
		T-MOTOR 7*2.4CF	50%	2	22.20	141	7750	6.35	50
			65%	2.9	32.19	198	9000	6.15	
			75%	3.9	43.29	252	10100	5.82	
			85%	5.1	56.61	305	11040	5.39	
			100%	6	66.60	338	11600	5.08	

Figure 6. Spec sheets for T-motor MN1806 KV1400