No Fly Zone / Collision Avoidance Drone

Nicholas Schoessling

Lewis University

**Abstract**

The No Fly Zone / Collision Avoidance Drone will be able to solve problems that new drone pilots face. Since drone flying is expanding to the consumer market, we solve two big issues that they face: If they fly in a no fly zone, then they could get fined up to $20,000, and most new pilots crash their drone into an obstacle. The collision avoidance system is powered with an Arduino Uno and a sonar sensor. The no fly zone system is powered with a Raspberry Pi 3b+ and the Navio2 flight controller. Code is stored on a software system called DroneKit. The drone takes into account of the ethical, environmental, and economic impacts it makes, as well as following several IEEE engineering standards. Testing the drone came with several issues which were thought around about in an engineering manner. Two crashes happened during testing with the final crash making the Navio flight controller unstable and unable to arm the motors to allow the drone to fly anymore. To fix this issue a new Navio controller will have to be swapped in.

**Introduction**

A rule that drone owners often overlook are of the no-fly zone rule. If we look at the company DJI, a popular company which specializes in making drones for photography, they have an entire page on their website dedicated to “fly safe geo zone”. This consists of detailed places where a drone cannot be flown, along with a map that drone owners can use to check if their drone is outside of the fly zone. Drones have been becoming more popular throughout the years. That means there are more owners, and more responsibility. We have decided to take on that challenge. Through this project we intend to build a drone that is automatically able to detect no fly zones and will not cross the boundary. The drone will also have a safety feature which detects objects, that way it will be a collision free drone as well. The reason we cannot use a pre-bought drone is because the software is secure and cannot be accessed by a customer. Since it is extremely time consuming to reverse engineer a pre bought drone and build sensors externally, we have decided to build it! In the rest of the sections of this paper, some background information will be given about what is going on. Next, the actual project will be broken down into the parts that were chosen and the process of building it. Then, the results of this project will be discussed. Lastly, a conclusion will be drawn from this project and possible future steps that could be taken.

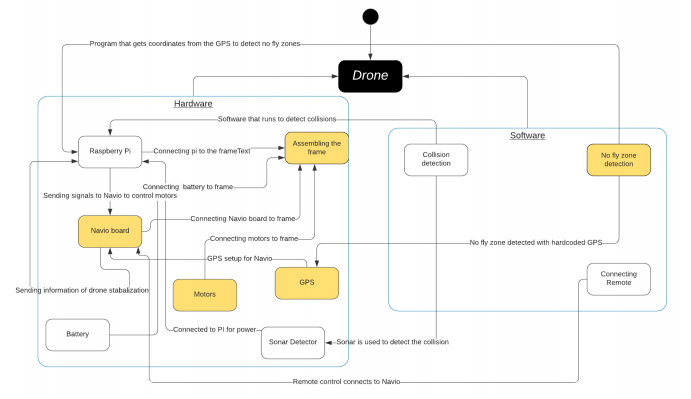
**Background**

Overall, we are going to make a drone that can detect collisions in all directions. We will be using HC-SR04 Ultrasonic sensors to detect obstacles near it. The drone is powered by a NAVIO controller which controls the motors to the drone. The rest of the drone is built off of custom parts such as the frame found on amazon. We will program the drone using a Raspberry Pi 4 or Pi 3b+. The stakeholders are very important because we would not want to cost them extra money. The FAA announced that if you are caught flying in a no-fly zone that you can be fined up to $20,000. The average consumer does not know that no fly zones even exist yet where they are. If they have heard of it, it might not have crossed their minds since they since want to focus on taking photographs instead of the no fly zones. Since the system will be able to detect if it is by a no-fly zone, they will not have to worry about whether they are in one or not.

We got this idea of building a fly safe drone from DroneRush.com. The “No drone zone: Common no-fly zones for drones in the US” article on their website states that the Federal Aviation Administration is charging up to $20,000 if a drone is in a no-fly zone. Some of these fly zones include sport arenas, wildlife refuges, forest fires, military bases, the White House, and the list keeps going. Through this background, we intend to engineer a drone that will not be able to enter any of the no-fly zone locations. And YES, we will register our drone with the FAA. In *Figure 1* from the FAA.gov, it states that any drone that weights between 250 grams (0.55 lbs.) to 25kg (55 lbs.) has to be registered under FAADroneZone registration.

*Figure 1.*

There are many different ways to approach building the drone. We will use a flight controller antenna which uses signals and systems (a course that we have taken) through a frequency of 900-1000 MHz. This connects to a RC controller. We will ensure that the four motors are sitting in a square pattern to help balance out the drone and maintain stability making sure the motors can carry the weight of the drone.



*Figure 2.*

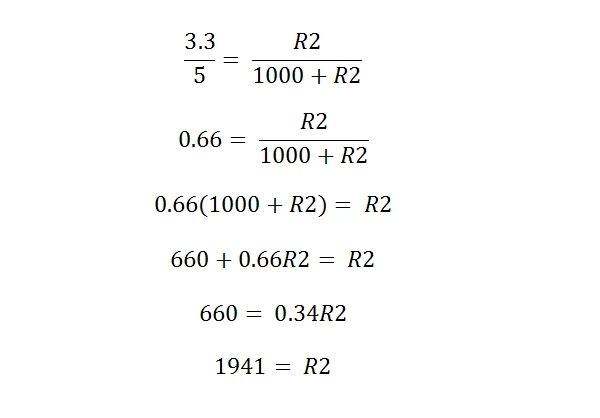
**Collision Avoidance / No-Fly Zone Drone**

Listed above in *Figure 2* is the block diagram that we worked with to demonstrate how the drone communicates with the different parts of itself. The two subsystems that were easily distinguishable are the no-fly zone and the collision detection modules. The no-fly zone subsystem is labeled in yellow, and the collision detection subsystem is labeled in white. The no-fly zone detection will come off of the GPS on the Navio board. Coordinates are able to be pulled from the GPS module and they can then later be hardcoded, so the drone knows where it is in space. The collision prevention system will come off the ultrasonic distance sensor connected to the Raspberry Pi. There will most likely be sensors in a 360 view around the drone. If there are 6 sensors around the drone (in front of, behind, to the left, to the right, above, and below) it would be able to detect how close the drone is from an object. To see if the drone will collide with something in a diagonal plane, it is possible to use two sensors and some basic trigonometry to calculate if it will hit an object diagonally. For all of the other parts there was no specific group they could fit in to go with the GPS module or the collision detection module, so they were just spread out between the two subsystems. The battery to power everything and the remote control to fly the drone were added to the collision detection subsystem and the frame that everything is built on and the motors were added to the no-fly zone system. With these two subsystems, the tasks were divided evenly being 5 tasks per subsystem.

The exact ultrasonic ranging module that is being used is the HC-SR04. This is an ideal sensor to be picked since it is one of the most popular. They are pretty cheap which helps since there will be 6 of them being used on the drone. They are pretty lightweight and also small in size being 4.5cm by 2cm by 1.5cm. Size and weight are important factors in drone building since if an item is too bulky or heavy there will be problems flying and controlling the drone. It has a pretty good range as well: being able to detect distances as far away as 4m up to a close 2cm. The 4m max range is essential since that will give us enough time to slow the drone down to prevent a possible collision with an item. The 4 pins on the sensor are a 5V power supply, 0V Ground, Trigger Pulse Input, and Echo Pulse Output. First a quick trigger pulse of 10uS TTL is sent out to let the sensor know to start the ranging. Then pulses of eight 40 kHz waves are sent out of the sensor raising its echo, and then it waits to see if any waves are sent back to the HC-SR04. If there are no waves sent back, that means that there is not an object within 4m to bounce the waves back. The echo is a distance object that is pulse width and the range in proportion [1]. In order to test the distance a quick calculation is performed. Test distance = (high level time \* velocity of sound(340M/S)) / 2. Another thing is when the pulses are sent out, they are not sent out in just a straight line perpendicular to the sensor. The pulses are actually sent out in a 30-degree cone. This cone shape can cover a lot of the corners of places that the drone needs to detect to prevent collision. The HC-SR04 is pictured in *Figure 3* below.



*Figure 3.*

Collision prevention from the HC-SR04 will be straightforward. Using the distance formula that is provided in the documentation, the drone can be split up into a few categories. Once the HC-SR04 is able to receive signals meaning that an object is within 4 meters, the thrusters will slow down to about 75%. When it reaches 3 meters, the thrusters will slow down to about 50%, 2 meters will be about 25% and when the drone reaches 1 meter it will completely stop going in that direction if the drone is still heading towards an object. 1 meter is a reasonable distance to completely . *Figure 4* stop the drone since you would not want it to be super close to the object and possibly have the drone still bump into the object due to wind or some other outside source. The drone will rarely have to go in close proximity to objects less than 1 meter apart since this is an outside drone. If an indoors drone is made in the future adjustments could be made. Only two imports need to be made, RPi.GPIO to add Trig and Echo to GPIO pins and time to keep track of time. In order to protect the Raspberry Pi there needs to be some resistors in the circuit. Since the ECHO port outputs 5V, we need to step down to 3.3V to prevent damage with a voltage divider circuit. [2] Using a 1kΩ resistor and a 2kΩ resistor this can be achieved. The math is shown in *Figure 4.*

These resistors would then be formed into a pull resistor combining them from the ECHO pin leading to a GPIO pin on the Pi, lastly connecting to ground at the end.

In order to even have any single piece of the drone working, there needs to be power to the drone. A good source of power that was found was through a lithium polymer battery. Lithium polymer batteries are really similar to lithium ion batteries except they have less protection. The cases are made out of soft polymers instead of a metal or plastic casing. Since weight is a huge issue for drones, we cannot have the drone have all that extra weight from the metal or plastic casing. This battery only weighs 248 grams, so it is overall pretty light for its size. They are pretty small and are able to fit right on top of the drone centered perfectly. Lithium polymer batteries are essential in drones because just like many other robotic machines require a lot of current. The specs of this specific battery are a 3000mAh capacity, 30C discharge battery. It contains 3 3.7V cells inside adding to a total of 11.1V. Lithium polymer batteries have special connectors to handle the amount of current that goes through these batteries. This one comes with a Dean Style T Connector and an XQ60 Connector. XQ60 connectors can support a total of 60A being drawn at any given time. T Connectors have a minimum of 60A but can go into the hundreds of amps at one time. T Connectors will most likely be used since it is possible for this battery to give off 90A of current. It would not be good if the connector could not handle the amount of current that was being released from the battery. The battery that will be used is attached below in *Figure 5.*

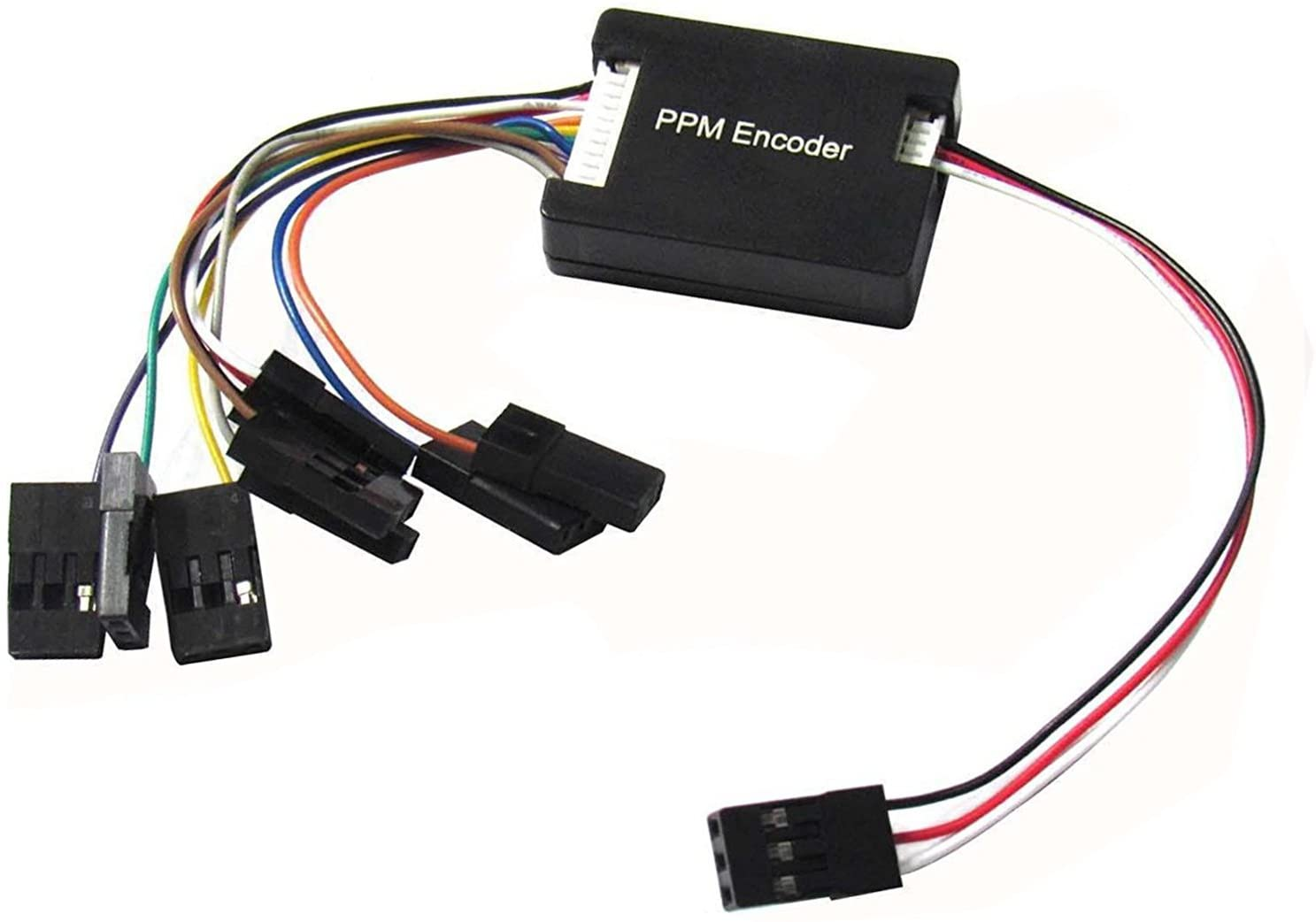


*Figure 5.*

The next part of the drone is how it is possible to communicate to the drone. Communication happens with RC and Telemetry. They are both used to wirelessly communicate with the drone. The main difference is that telemetry is bidirectional, and RC is unidirectional. With bidirectional the drone can communicate with the device on the ground and the device on the ground can communicate with the drone. RC will be using a 2.4GHz band and the telemetry will use a 915 MHz band. RC controllers are usually pretty expensive and the more channels they have the price goes up exponentially. Most drone remotes need 6 channels so this remote has 6 channels. 4 of those channels are dedicated to the roll, pitch, yaw, and throttle. A fifth channel that can be added could be a switch for an autonomous mode if someone would like to have the drone follow a set path or just manual mode. The 6th channel is still free and there will probably be a use for it with more research. Attached below in *Figure 6.* is the remote that will be controlling the drone.



*Figure 6.*

RC is powered with pulse width modulation (PWM). This remote’s value goes from 1000 PWM to 2000 PWM. Different signals give off different PWM’s. In order to send all of these signals to the NAVIO board there needs to be a PPM encoder. The PPM encoder condenses the 6 channel’s PWM values into one signal which is sent to the NAVIO. This one comes with 8 channels even though we only need 6 channels. 8 is the standard for most PPM encoders so having more is not hurting the drone. It is always possible to add an 8-channel remote later on if there is a need for these extra channels. The PPM encoder is shown below in *Figure 7.*

With RC being used for the direct control of the drone itself, there is telemetry to return data to the user. Some data such as location, speed, battery life left etc. is returned to the user. This information is vital for us to know so we know what is happening to the drone when it is airborne. The telemetry device is plugged into a computer and the other piece is plugged into the drone. As seen in *Figure 8.* below the telemetry . *Figure 7.* device is super light only weighing 4 grams without the antenna. This barely adds to the weight so we can still receive all this data. One restriction that this leaves us with is that the drone needs to be on a network when it is flying. This is not really a big problem at Lewis for example since the campus has a Wi-Fi network all around school, however this can lead to problems if no network is available. One possible solution is to attach a mobile hotspot to the drone and constantly have the drone communicate on that network with the computer.

*Figure 8.*

Some aspects about this project to consider are the various impacts it may have on the world whether that being ethically, environmentally, or even socially. One big environmental factor with this project is the use of a lithium polymer battery. Lithium batteries in general are super volatile and it does not help that this is a lithium polymer battery. Since the battery cells are encased in a thin sheet of carbon, it is really easy to puncture them. Lithium is a part of the alkali metals and this group on the periodic table is the most volatile, having them create huge explosions if they react with something. Lithium is really great at storing energy. When it is released as a trickle, it powers your phone all day. When it is released all in one go, the battery can explode. [3] One thing we would recommend customers to have is a lipo safe fireproof bag. When the battery is not being used, it is best to store the battery in one of these for safety reasons.

There is also a major economic impact to consider with this drone. With where we are at now in time, drones are still expensive to produce for the masses. If you want a kid’s toy, one could spend $50 on the drone but it will not last too long. A basic drone costs about $200. The Navio board on this drone costs $200 by itself. This drone costs about $400-$500 for us to build it. A drone this quality would probably cost somewhere around $800-$900. Companies might be more interested in this drone than an average consumer since their drones would be more expensive because they can budget more money for them.

Another major concern in the ethical field with this drone would be privacy. Drones of all shapes and sizes could be a huge breach of privacy if they are in the wrong hands. Someone could get a drone and use it to spy in residential areas by looking through people’s windows. There is not much that can be done to prevent this from happening to our drone however government sectors will be protected. One branch of the no-fly zones is that all important buildings are restricted no fly zone areas with no exceptions. This includes power plants, prisons, and places like Washington DC. This is good because our drone will not be used to aid people in prisons. Earlier this year there was a drone that was used to drop off drugs into a prison in Belgium. [4] The drone fortunately crashed but with ours they will never have the chance to go anywhere near a prison.

One last impact is by limiting the speed of the drone itself. It is possible to have the drone only go max speed if the altitude is high enough. It would not be smart to drive a drone 40 mph when it is gliding across the floor, many people can get hurt from this. With the telemetry device, the altitude gets returned to the user as a data set. When this data is received the max speed allowed on the RC controller would be limited to a set amount for safety. Max speeds will be hardcoded to a certain altitude. This would help younger users since they might not be a good drone pilot due to their age.

Engineering standards are important to take note of when designing and creating projects. There were a few standards that were looked into when developing this project. As talked about earlier, lithium batteries can be very dangerous which is why our batteries will be going under careful inspection to make sure they are safe and follow IEEE 1679.1-2017 standards. [5] This standard talks about the characterizations and evaluations of lithium-based batteries in stationary applications. Another standard goes over the drone’s motors. We need to make sure that we have safe motors that keep the drone up and safe from crashing when airborne. This standard is the IEEE 3004.8-2016 standard. [6] This goes over the protection of motors used in industrial and commercial power systems. There is actually a standard that goes over the airspace for UAVs. A UAV is an unmanned aerial vehicle which is technically a drone. You cannot fly all over the place all recklessly when you could hit somebody on the ground. This standard is IEEE 1939.1. [7] One last standard goes over communications between the user and the drone. It is possible to interfere with other aerial vehicles with your own signal, so it is best to follow the procedure of making sure to stick to the right channel. The standard is IEEE P1920.1 [8] which resembles everything related to aerial communications and networks.

Working with the subsystem was not an easy task at all. At first there were theories where it would be possible to hook up the sonar sensors to the raspberry pi, however none of those theories were true. The Navio takes up the pins of the raspberry pi and if we try to use those pins then the Navio will not work which we need to fly the drone. Navio does have some pins available on the side of it, however we do not have enough. I had to think of something else to run the sonar collision detection system off of. There were a plethora of Arduino uno’s available, so I attached the uno to the drone. Looking for places to put it, I decided to extend the top frame by adding some spacers and placing the top frame above the Navio and raspberry pi. The pins are connected from the sonar sensor to the Arduino which is then connected to the Navio. The battery is hooked up to the power distribution board which then powers the electronic speed controllers, and then the motors. It is stepped down from 12V to 5V for the Navio to be powered. The RC controller is hooked up to the RC receiver which is then transferred to the ppm encoder and lastly to the Navio to run the RC commands.

In order to test this subsystem, there are a few tests that must be made. To test the RC controller, we can go into mission planner which is a software that gives us all the information we need about the drone. One of the tests that you have to calibrate in mission planner is Radio Calibration. In this test, I would test the 6 channels that I have allocated. I would move the left and right sticks in all 4 directions to see the minimum and maximum throttle, yaw, pitch, and roll bars go up and down. Then I will test the switch, and potentiometer on the controller. The switch is for a stabilize mode to allow the drone to stay at one altitude in the air. The potentiometer will be used to switch between different modes. There are three modes: ones being all the way left, one all the way right, and one in the middle. The left is to allow you to use the remote when you are flying the drone. The right is for the drone to return and land automatically. The middle is a mode that you could program such as the drone flying in a circle. If I see all the bars move, then the controller is functioning properly. In order to test the battery, I will see what happens when I plug in the drone. When you power the drone on, you will hear beeps from the esc’s saying that they are arming. Then the LED light on the Navio will flash red and blue saying that it is starting up. Then if the remote is off it will blink yellow. Once the remote is on, it will become either blinking green, or blinking blue depending on how well it can connect to the GPS. If all components of the drone are able to receive power by seeing LEDs on them or actual functions such as the motors spinning, we know that the battery has worked. In order to test the sonar system, we can hook it up to the Uno and go to the terminal. I have the Uno print out to the console the current distance an object is away from it in cm. If you get too close to the drone it will tell you that you are too close to an object. The sonar sensor updates every 12ms so it is really accurate, and you would not have to wait a long time to see if it is working. I am going to place an object such as a box in front of the sonar sensor. Then I will slowly move the box closer to the sensor watching the console seeing the numbers count down. Once I get close enough to the drone, I will see the console saying that the drone is too close to an object and that it is stopping the drone.

All throughout the project safety has been a number one priority. All the different parts put together into the drone had a lot of thought in it for this subsystem. For one: the battery is one of the most dangerous parts. With this being a lithium polymer battery, it is highly flammable and explosive. Being polymer and not plastic, it is pretty easy to puncture. The battery has been securely tightened down with Velcro straps to make sure it will not fall out when the drone is flying. The frame protects the battery from getting damaged. Another way for the battery to explode is if you were to overcharge it. With the balance charger, certain settings have to be entered in to make sure that the battery will not be damaged. The balance charger is set to charge at a rate of 1.5A and charge up to 12.6V. The battery has two connections: one directly to the charger, and the other to the balancing port in the charger. If the battery is not balanced, then voltage will be distributed unevenly in the 3 cells. When the battery is being charged, it is placed in a fireproof box just in case something catastrophic does actually happen. It is also never left alone to be charged. While it is being charged someone has to be in the room. Note: In *Figure 9*, the battery is not placed in the fireproof box for demonstration purposes only.

Another safety feature is built into the remote control. When the drone is being set up and is flown in stabilize mode using the remote control, it needs to be properly armed. If you turn the remote on and the throttle stick is up, the drone will not arm, and an error message will display on the remote saying that you need to reset the remote control and set the throttle back down to zero and all other channels set to zero. Once the remote is set to zero the drone needs to be armed. To do that, the throttle stick is pushed down and to

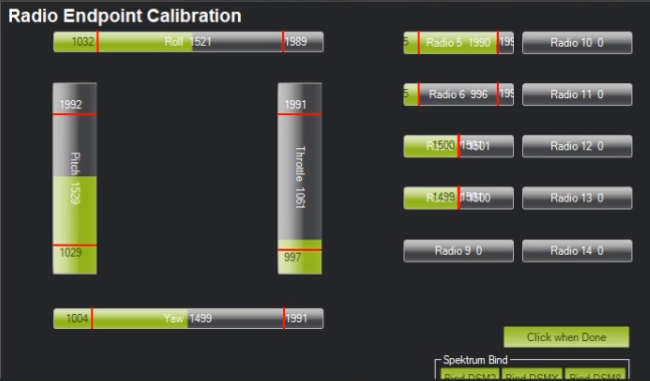
the right for 5 seconds. This is to

prevent the drone from accidentally

*Figure 9*

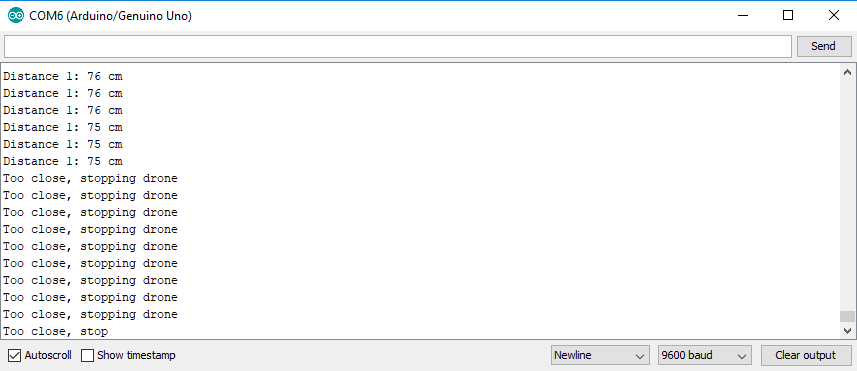
being taken off if the throttle stick is accidentally moved. To turn the drone off, a similar action is done. The throttle stick is moved down and to the left for 5 seconds. Both of these two systems help with safety in the subsystem.

Testing the subsystem took many iterations to get right. The battery and the RC remote were really easy to test and they all functioned right. Going through the setup for the radio in *Figure 10* was simple. All 6 channels that are allocated to the drone work. In the diagram you can see channel 7 and 8 active at 1500 because we have an 8-channel ppm encoder. On the drone, those two channels are not plugged into anything. They are the gray and brown wires. If we had an 8-channel remote, we could connect those two channels to the remote. With



*Figure 10*

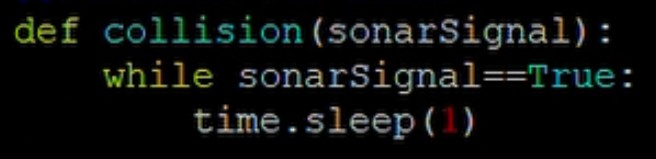
Testing the battery was easy since everything turned on and lights were on for each part that has a light. The sonar sensor system was the most challenging part to integrate. As stated earlier, it was not possible to test on the raspberry pi directly, so another microcontroller had to be involved with this project. The code for the program is attached to this report. [9] Converting from Python to C++ was not a hard task since I have used Arduino’s before. When I tested the sonar sensor and I went to the serial monitor, I saw that it was working. I put a box 100cm away from the sonar sensor and it was saying that it was 100cm or 99 cm away which is pretty accurate. I slowly moved it closer to 75cm which is what I set the triggering distance to. Once it reached here it notified you that it was stopping the drone and that you were too close according to *Figure 11.*



*Figure 11*

The next steps of this subsystem are to integrate it into the entire system. A little work has to be done to integrate the Arduino into the drone. We need to find proper placement for the Arduino to mount on the drone. Since we have an extra frame set, we used spacers on top of the top frame to mount another top frame to house the Arduino. To attach the Arduino there will be double sided sticky tape attached to the frame. The wiring will be more complex since the propellers cannot be hitting the wires tangling them. In order to prevent this, we will bind the wires close to the frame, so they are not free. In order for the sonar sensor to not get bad results from interference from the blades, we will mount it below the drone. The drone will be placed on stilts so it is elevated, and parts can be placed below the drone. For the collision detection to work, a signal needs to be sent from the Arduino to the Navio. This signal is set to high from digital pin 4 on the Arduino when the sonar sensor picks up a distance of less than 75cm. This connects to the other subsystem by using DroneKit. DroneKit is software where you are able to program with your drone. Using DroneKit we will be able to stop the drone from moving if that signal is set to high. This will be a lot simpler than just slowing it down like how I originally planned to do it.

**Results**

Connecting both subsystems had a few different connections. To complete the connections the Arduino had to be connected to the Raspberry Pi / Navio bridge. This was connected through a singular wire. The Arduino had a pin which was declared as digital pin 4. This wire then followed through to a side pin of the Navio board. With the Arduino detecting if the drone becomes within a distance of 75cm to an object then the signal will be set to high. This high signal is then picked up through our drone kit program. In the drone kit program depicted in *Figure 12* the function for collision detection goes through a while loop that detects if the signal is high or low. If the signal is high, then the drone will stop in its path. Once it switches back to low, the drone is able to move again. This program is a part of the no-fly zone program, so they interact throughout the software.

*Figure 12*

To connect the remote to the system, the remote sends out a signal of 2.4GHz to the RC receiver. The RC receiver is then wired to the PPM encoder. The 6 channels in that the PPM encoder receive are then encoded and sent to the first row of external pinheads. Next to that pinhead are the 4 ESC wires. The ESC’s are zip tied underneath the frame twice to make sure that there are no lose wires that can be tangled. The ESCs are then wired to the motors which are attached at the end of each arm. The other end of the ESCs is soldered to the power distribution board in the bottom frame. For this whole system to have power the battery needs to be connected. The battery is Velcro strapped on top of the bottom frame. This is then connected through a deans to xt60 adapter to fit the power module. The middle of the power module is connected to the Navio board for the Navio, Raspberry Pi, and Arduino to receive power. The rest of the hardware components are then connected to the power distribution board for it to receive power. For the drone to be able to detect where it is in space, we have a GPS module fixed to an arm of the frame connected to the Navio. To receive the rest of the data on a computer we have the telemetry module wired to the Navio board connecting through Wi-Fi to the other end of telemetry plugged into a computer.

One safety strategy that came to mind to complete the subsystem was to make sure that any wires on the drone were secured. If a wire would get caught in the propeller then it would most likely crash due to being tangled or cut from the blade. To fix this issue we had to wire the wires through the frame to make sure that there was no slack. If there was still some slack on a wire and it might impede the blades, then we fixed the wires to the frame with a zip tie. Another safety measure that we found out a little bit late was to change out the spacers that we have in the drone. The original ones are made out of nylon which is really common for spacers. Since the drone goes under stress when flying, they can snap easily which is what happened during the first crash of the drone. A tougher version of spacers that could be used are made out of brass as depicted in *Figure 13*. They are a little bit more expensive, however they are worth it for its strength.



*Figure 13*

Results integrating the system came out as expected with the functionality working except for the drone itself. The drone did not want to fly anymore. After the first crash when we changed the spacers out, we investigated what could cause the drone to crash. We determined that an ESC was at fault. The ESC would get hot and it was not spinning as fast as it should be to maintain stability. In order to figure out why it was getting hot, we made sure that the soldering points were soldered correctly on the power distribution board, and that both wires were fully connected which they were. We decided to cut the heat shrink open to see if there are any problems internally. This is where we found our issue as shown in *Figure 14.* As pictured

**there is a dent and a scratch in the capacitor. This was a faulty ESC that we received so more were ordered to replace it. This fully fixed the issue when a new ESC was soldered to the board. The drone crashed a second time however which brought more complications. The back plate of the cover came off of the RC remote and a battery got knocked lose. Putting the battery back in the drone created a power surge which sent the drone flying and it crashed once

*Figure 14*

more. This crash brought more problems to the project. A propeller snapped in half which was easy to exchange for a different one. When the drone was booted up, there was a faint green light on the Navio board. To check on the raspberry pie, we tried SSHing into the drone, but no connection could be made. Diagnosing this issue, a slightly damaged wire was located. The power wire had a slight discoloration which was left from a blade hitting it. Power might not be getting through to the board, so the wire was sliced and replaced with a new wire however the issue persisted. The operating system might have been corrupted from the crash so a fresh image was installed on the raspberry pie however it still would not connect. Upon closer examination in *Figure 15* a chip on the raspberry pie was chipped. The left pi is the chipped one, and the right pi is the new one. A spare raspberry pi 3b was in the lab and was replaced with all the software reinstalled on the drone.



*Figure 15*

However, the drone still had issues even after the Pi was replaced. The new Pi was a Raspberry Pi 3b instead of a Raspberry Pi 3b+, but the only difference is the processing speeds. The drone will still function the same on either model of the Pi. SSH functionalities were restored with the new Raspberry Pi and we were able to install DroneKit and our programs on it. When we were getting closer to taking off, the drone was having trouble arming the motors. This is the final step before the drone can take off. A double blinking yellow light came off of the Navio which means that the motors will not arm for safety reasons. To test this further, individual motors were tested to see if the motors can be forced armed. For safety reasons, the propellers were taken off of the drone and one motor was tested at a time. Each motor responded to the RC controller fine so the motors were not an issue. Upon further research through forums and videos the conclusion that we have come to is that the Navio board became unstable. When the drone crashed, the Navio controller was damaged and would need to be replaced. The ESCs are able to detect that the Navio is not functioning as it should be and will not let the motors spin all at once. An internal library most likely has been damaged on the Navio. To fix the issue of the drone not being able to fly, this will need to be replaced. However, due to the time constraints of this project, the Navio will not arrive on time. The core subsystems are functioning as they should be however, the drone will not be able to fly.

**Conclusion**

Even though the drone in the end is not able to fly with the motors not arming, the subsystems of what we tried to accomplish are fully functional. The collision detection system is only able to detect one position of the drone, which we have being the front of the drone. It detects whether an object is within 75cm of the drone and once it hits that, the drone will not move any further. The code is finished on the Arduino which is coded in C++. This is then transferred to the Raspberry Pi and is in a function coded in Python. The No-Fly zone program is also on the Raspberry Pi and it uses DroneKit. Using a program called Mission Planner, we can see the drone’s real time coordinates, and a bunch of other data on the drone. Using the drone’s coordinates you are able to build a box around the drone and it will not fly beyond the box. If we could have done something different, we probably could have prevented the second crash. Since the drone took off on the table we could have disarmed the motors and then put the batteries back in the remote control. We could have also put the drone on the floor so it would tip over instead of fall. Another thing we could have changed was to do more research initially and find out that certain things such as incompatible programs or propeller sizes matter. This would have saved a lot of time in the planning phase and by now the drone might have been finished. In the future, if we replace the Navio2 flight controller, the drone should be stable and be able to fly. Our programs will work and the product will be finished. If the new Navio2 board does not fix the problem, then we shall use engineering principles to find another possible issue for why the drone is working. This is the engineering process.

**References**

[1]

“Ultrasonic Ranging Module HC - SR04 ,” *Ultrasonic Ranging Module HC - SR04*. [Online]. Available: https://cdn.sparkfun.com/datasheets/Sensors/Proximity/HCSR04.pdf. [Accessed: 28-Sep-2020].

[2]

“HC-SR04 Ultrasonic Range Sensor on the Raspberry Pi,” *The Pi Hut*, 22-Oct-2013. [Online]. Available: https://thepihut.com/blogs/raspberry-pi-tutorials/hc-sr04-ultrasonic-range-sensor-on-the-raspberry-pi. [Accessed: 28-Sep-2020].

[3]

H. Guinness, *Why Do Lithium-Ion Batteries Explode?*, 16-Jan-2018. [Online]. Available: https://www.howtogeek.com/338762/why-do-lithium-ion-batteries-explode/. [Accessed: 28-Sep-2020].

[4]

C. Tomlinson, *Drone Delivering Illegal Drugs to Belgian Prison Crashes*, 16-May-2020. [Online]. Available: https://www.breitbart.com/europe/2020/05/16/drone-delivering-drugs-to-belgian-prison-crashes/. [Accessed: 28-Sep-2020].

[5]

M. Nispel, “IEEE 1679.1-2017 - IEEE Guide for the Characterization and Evaluation of Lithium-Based Batteries in Stationary Applications,” *IEEE SA - The IEEE Standards Association - Home*, 01-Jan-2018. [Online]. Available: https://standards.ieee.org/standard/1679\_1-2017.html. [Accessed: 12-Dec-2020].

[6]

R. Rifaat, “IEEE Std 3004.8 - 2016,” *IEEE 3000 Standards Collection*, 07-Dec-2016. [Online]. Available: https://mentor.ieee.org/3000-stds/dcn/18/stds-18-0003-00-PUBS-3004-8-2016.pdf. [Accessed: 12-Dec-2020].

[7]

X. Chenchen, Ed., “IEEE P1939.1,” *1939.1 - Standard for a Framework for Structuring Low Altitude Airspace for Unmanned Aerial Vehicle (UAV) Operations*, 17-Aug-2020. [Online]. Available: https://sagroups.ieee.org/1939-1/. [Accessed: 12-Dec-2020].

[8]

“Use of Cellular Communications to Support Unmanned Aerial Vehicle (UAV) Flight Operations,” 2019. [Online]. Available: https://access.atis.org/apps/group\_public/download.php/48760/ATIS-I-0000074.pdf. [Accessed: 12-Dec-2020].

[9]

N. Schoessling, “nschoessling/DroneFinalProject,” *Drone Final Project*, 11-Sep-2020. [Online]. Available: https://github.com/nschoessling/DroneFinalProject. [Accessed: 12-Dec-2020].