

**Framework for Autonomous Delivery Drones**

**School of Applied Technology**

***ITMT 495***

**Professors:**

**Jeremy Hajek**

**Dr. Dan Tomal**

**Students:**

**Alexandre Mutti Fábio Monfardini**

**Abstract**

Many types of online purchased merchandise are light in weight and small in size.

The distance between the online seller and the buyer, and the absence of delivery methods at some stores like pharmacies show potential for using drones for package delivery.

Shipping trucks can carry a large quantity of small light packages, but can’t be used to ship only a single package, and they increase the severity of traffic congestion. Other methods such as car delivery service are only offered within certain industries, but not by pharmacies and governments. Another important factor is the necessity of asphalt roads for previous methods to perform their task, which prevents them from working in areas that lack these types of roads, like some third world countries and post-disaster areas. Quadcopters have been proposed to carry and deliver packages that are light and small. They can be controlled by a remote control, or they can use autopilot. Here we investigate the impact of using autonomous drones to carry packages between several geographical locations, and the reliability of the automation system in terms of wind resistance, GPS accuracy, and flight stability. We use the 3DRobotics Iris quadcopter to test and build the automated system.

**Table of Contents**

**Chapter 1: Introduction**

**Chapter 2: System Overview**

**Chapter 3: Hardware Components**

**Chapter 4: Software**

**Chapter 5: Experimental Work**

**Chapter 6: Results and Discussion**

**Chapter 7: Conclusion**

**Chapter 8: Scope For Future Work**

**Chapter 1**

**INTRODUCTION**

A rising demand for flexible and efficient forms of delivery that utilize renewable energy resources and minimize on human impact, especially traffic, has recently called attention to the possibility of using autonomous unmanned aerial vehicles (drones) in a widespread commercial application. Many organizations worldwide, from retail outlets to national governments, have expressed interest in utilizing this technology for diversifying their fleet. Additionally there are many opportunities to implement drones to reach areas that were previously too costly or too dangerous to deliver supplies to. In light of these opportunities both commercially and socially, we set out with the goal of developing a vehicle that could meet these demands.

The drone vehicles currently available come in a wide variety of configurations depending on the features, power, battery life, and vary in price and quality. Due to their nature structural integrity and modularity is an essential factor, and the manufacturer we selected, 3D Robotics, is known for meeting these two requirements. The model selected was the Iris quadcopter, which is a sturdy all-inclusive vehicle with autopilot, radio telemetry, manual controller, and Global Positioning System (GPS) included in addition to all of the necessary hardware components, and a payload capacity of 400 grams, which is sufficient to the scope of this project.

These vehicles would have to be able to maneuver themselves between at least two locations carrying a payload as commanded by a remote user (the destination). This includes autopilot, mission planning, and user-interface software, in addition to the 3D Robotics Iris vehicle, server, and radio telemetry hardware required. Success would entail accomplishing automated navigation to and from a destination without human intervention, and would require the development of server side software for automating the pre-loading of navigation specifications, pre-arm checks, and launch of the vehicle.

The project was divided between hardware and software in an effort to capitalize on strengths and maximize on the short amount of time allotted. The software portion addressed determining flight paths, methods of communication, and automating the flight process utilizing pre-existing programs where possible and developing tie-ins to allow for remote user requests. The hardware portion focused on mastering the actual flight of the drone, calibrating sensors and motors, telemetry, navigation systems, mastering drone parts assembling and disassembling, configuring RC transmitter, and drone repairing. Through a series of agile iterations the research, development, and testing cycle moved from first assembly of the drone and testing initial functions, through thorough flight and process analysis, to ultimately successfully conducting remotely initiated automated flight and return.

This report includes sections detailing our research and findings, and future prospects for further development. Chapter 2 is an overview of the systems involved, and briefly explores the process necessary to a successful trip. Chapter 3 is a list of the hardware components included in the Iris vehicle and their function. Chapter 4 is a discussion of the software used and developed for this project, how it communicates, and its function. Chapter 5 explores the experiments conducted during the duration and their content. Chapter 6 discusses the results of the project and what was discovered. In Chapter 7 we conclude the project, and finally in Chapter 8 we present future possibilities for further development.

**Chapter 2**

**System Overview**

In this chapter we’ll explain how the system is set up in order for the drone to deliver the package. The architecture of the system is as shown in Figure 2.1 below.

We developed a code that can gives direct commands to the drone’s auto pilot, we used the DroneApi-Python provided by 3DR. This API creates a connection using MAVlink, that provides resources to access the connected vehicle telemetry, state and parameters, enabling both mission control and direct control over vehicle movement. The API provides libraries an “easy” way to code in python for drones, many commands are pre-defined using libraries. The instructions that will be sent to the drone will achieve the following process: Taking off, reaching a specific height, start moving towards the destination where the user is, reducing altitude, dropping the package and then using the RTL mode (return to launch) that is pre-defined in Pixhawk Auto Pilot

In order to perform a successful trip, the following factors must be available and achieved:

1- GPS Lock on the drone: Prior to flight, the drone must have a GPS lock to make sure it knows where home station is so it can come back to it after the package gets delivered. A green light should blink on the drone to indicate that GPS lock is successfully obtained.

2-Preflight check: A usual physical procedure that should happen before every flight. The procedure includes checking all the locking nuts above the propellers to ensure they are tight enough. Setting the antenna in a vertical position to maintain the best connection quality between the drone and the ground station. Making sure the drone is standing over a level ground. Arming and disarming the drone using the RC transmitter. Taking it off to up to a low altitude (2-5 meters) then landing it. Checking that all blades are spinning equally when taking off. And finally, checking that the path is clear for taking off.

3-Connectivity check: After performing the preflight check, then we establish the connection between the ground station and the drone and make sure they’re successfully talking to each other.

4-Turning off safety mode: This is the final step. After performing the previous 3 steps, then we’ll physically put safety mode off by pressing on the red button on top of the drone until the blinking red light on it turns solid, which indicates that safety mode is off. A confirmation tone will be heard too.

**Chapter 3**

**Hardware Components**

We’re currently using the 3DRobotics IRIS+ quadcopter for prototyping. This quadcopter uses a PIXHAWK microcontroller to control the entire system of the drone.

The list below includes all the items of which the drone is built from:

1- A Set of 4 Arms (2 black, 2 blue): The rear arms have a wider angle with the drone body than the front arms.

2- A Set of 4 Propellers: Two of them are built to rotate clockwise, the other two are built to rotate counter clockwise.

3- A Set of 4 Legs: The 4 legs have the same design, but they differ in color for visual purposes. We have a set of 4 short legs and a set of 4 tall legs.

4- Four Motors: Two of them are set to rotate clockwise, the other two rotate counter clockwise. Each two have appropriate propellers connected to them.

4- Drone Body Shell: A plastic case containing the following parts:

* PIXHAWK microcontroller with a MicroSD card
* Power Board that connects to the 4 motors
* Lithium Battery
* GPS Module
* RC Receiver
* Speaker

5- RC Transmitter

6- Ground Station: A 915 MHz connector that can be connected to a laptop or a tablet. This is device that is used to connect between the home station and drone during the flight.

Fig 3.1, the 3D Robotics Iris Quadcopter with included parts (http://www.adafruit.com) 

**Chapter 4**

**Software**

The Iris+ itself utilizes the 3D Robotics PixHawk autopilot controller and the PX4 autopilot firmware. This communicates with ground station software via the MAVLink communications protocol. In researching a way to communicate with the Iris from a computer we discovered many implementations of software ground control stations, many of which were useful to configuring the drone, but MAVProxy stood out as the most useful to our case.

**Chapter 5**

**Experimental Work**

Different experiments were performed to insure success of each part was achieved.

**1. Calibration**: The IRIS+ quadcopter need several calibration procedures to be able to fly properly. The importance of this part is extremely high if not the highest. Compass and Sensors calibrations can be done using the MAVLink software or a terminal. The ESC calibration is done using the RC transmitter.

* 1. *Compass Calibration:* Involves moving the drone around all axiss. This will calibrate the drone to move correctly in the 4 directions.
  2. *Sensors Calibration:* Involves positioning the drone on its left side, right side, nose up, nose down, and on its back. This is important for the drone to be stable while flying, and to have a correct reading of its angle with the horizon.
  3. *ESC Calibration:* Using the RC transmitter, we follow a procedure of connecting and disconnecting the battery three times to calibrate the 4 motors and make them work with complete synchronization.

**2. Testing the multiple flight modes:** Using RC Transmitter, we tested Stabilize mode, Altitude mode, RTL (Return to Location), and observed how the drone reacted to these modes. We also observed how smooth the transition was between several modes while flying.

**3. Testing MAVLink:** We used the Mission Planner software to fly the drone, observed how it reacts to the pre-flight plan.

**4. Testing the code:** Before loading the full code into the drone, we tested several times each part of the code to ensure the code is working properly. We performed the commands take off, go to a specified height and land many times. We tested the method to reduce height when the drone reaches the target waypoint and then tested the code with all the procedure to make the drone delivery a package.

**Chapter 6**

**Results and Discussion**

The ongoing process of the project taught us a lot about autonomous flight control systems, drones aerodynamics properties, battery power, physical threshold of the equipment, and real world applications management.

We’ll list the results we’ve obtained then a discussion will follow.

**Results:**

1. Wind turbulence is a major obstacle. It can lead to crashing the drone very easily in a blink of an eye. It should be taken very seriously.
2. Accurate Calibration is extremely important. Inaccurate calibration could be a deal breaker. The Iris+ requires a frequently recalibration of its motors, we did it every time when swapped a battery. The instructions to be done are in this link: https://www.youtube.com/watch?v=sKBVadRDcEw
3. Frequent checking on drone settings and RC transmitter is necessary.
4. We can use PIXHAWK to control the extra sensors and the package delivery mechanism in the future, which is better than using an additional microcontroller such as Arduino or Intel Galileo.
5. In reference to our flight time and past experiments, it’s reasonable to state as fact that the current battery power we have, is enough for the drone to safely perform a round trip over a distance that is up to 5 Kilometers with no package mounted, and up to 3 kilometers with a package mounted.
6. Automation algorithm can be generated and fed to the drone once only prior to flight. After that point, the drone doesn’t need any further instructions from the server to perform its trip, and it can fly totally on its own.
7. If a trip was planned over spaces, where the drone can fly freely with no possible obstacles in the way, it displays excellent performance without the need for any additional sensors.

**Discussion:**

1. **Wind Turbulence:** When testing drone resistance against the wind in the windy city, you learn a lot about wind turbulence. Flying the drone in an open space will significantly reduce the chance of crashing in a tree or a building because of the wind. In this scenario, wind turbulence will affect battery power. The more wind fights the drone against the direction it’s flying towards, the more battery consumption the motors will need. When flying the drone in a windy weather, the sound of the motors increasing their power to resist the wind was heard loud and clear. This result means that even when a mission is planned over open space, the wind condition will affect the battery life which means the trip should be studied carefully in terms of the distance to wind power ratio to ensure it can be done.  
     
   Now when there are obstacles in the way, especially trees with empty branches, flying on low altitudes is not recommended. The risk of crashing the drone gets very high when flying next to a tree on a low altitude, one wind blow is enough to move the drone towards the tree. If that happens, the possibility of a crash is 100%. Whether the drone was controlled with the RC transmitter or the server (Autopilot Mode), and since the drone is very sensitive to the RC transmitter sticks movement and also very powerful, trying to fly it away from the tree is very difficult and tricky, and in such a short time (seconds) there won’t be a high chance of flying it away from the tree or any other obstacle.
2. **Calibration**: When calibrating the sensors of the drone using MAVLink, a very slight tendency could cause an inaccurate calibration. Even if the drone is only 1 degree inaccurate, when it flies and tries to maintain its altitude, it will drift toward one of the 4 directions. It will also lead the drone to flip when taking off causing it to crash instantly afterwards. In the drifting scenario, trying to stabilize the drone using the RC transmitter is challenging, and a manual trimming procedure, using the 4 trimming functions on the transmitter, should be done while the drone is in air. Calibration errors can lead to unexpected flight errors and possibly crashes, therefore the procedure of calibration should be done very carefully.
3. **Frequency of Check Ups:** When more than one person is working on the drone, like in our case, connecting the drone to more than one laptop might easily change some values for important parameters in the drone. That could happen due to different configurations on each laptop, and this will lead to errors in flying it.   
   Therefore, to avoid this scenario from happening, it is recommended to frequently reset the settings on the drone back to the default values, to make sure every value is exactly what it should be to perform a successful flight. A good communication between team members is also recommended to keep all members on the same page regarding the drone settings.
4. **PIXHAWK Abilities:** We noticed that PIXHAWK has many free unused pins on it, which creates the possibility of connecting more sensors in the future to it directly, instead of using an additional device, which would make communication between PIXHAWK and the extra device a complicated and unnecessary task.
5. **Flight Plan Distance:** The Iris quadcopter can reach a flight speed of 30 [Mph]. Although in our tests and experiments we didn’t fly the drone outside a circle of 100 meters radius, where we would be standing at the center of that circle, but the speed of the drone was high enough to estimate the distance it can cover with the current battery power.
6. **Automation Algorithm:** Since we only need to feed the drone flight instructions only once prior to flight, this gives the chance to focus on the real time readings from the drone while flying to assure no errors are happening.
7. **Open Distance Advantages:** When the drone receives a mission to fly in an area with obstacles (Buildings, Trees, Street Lights, etc.) in the way, without the ability to sense these obstacles, it’s impossible to accomplish the mission without raising the drone to an altitude that’s higher than these obstacles. Adding sensors in the future will enable the drone to maneuver between the obstacles. Therefore, in the current state of the drone, flying among open spaces is achievable without the need of any additional technologies.
8. **GPS lock problems**: We noticed that the Iris had some GPS issues. The Drone will not fly until it gets at least 5 GPS satellites, but if IRIS+ lost GPS flying, is the worst situation when flying the drone. Somehow when GPS lock is lost, the Drone loses all its orientation and start spinning in the air. The preset command in the Iris is to land immediately when it loses GPS lock but this is not happening when we load our code into the drone. A supposition is the lack of GPS lock is affecting the drone’s compass and that makes the Iris start spinning without orientation.

**Chapter 7**

**Conclusion**

The experiments we conducted on the drone, and the results of these experiments prove that the drone can be efficiently used to carry a light package that weighs no more than 400 grams over a distance of 3 kilometers. We do not had enough time to plan a big mission, but we tested successfully short missions in the soccer filed at IIT. Prior tests with the drones proved it is capable of flying in a 3km radius and around 20 minutes of flight with a single battery charge. As the nickname of the city says, the windy of Chicago could be very dangerous for the Iris+, it can cause crashes easily and should be used carefully in windy days.

**Chapter 8**

**Future Scope**

The use of drones for package delivery is highly promising, and within a couple of years, when they prove to be a reliable technology for package delivery, they will be globally used for that purpose.

Among the many features that could be developed, we list those who have the highest priority for a safe and reliable package delivery.

1. **Sensors:** To make the flight plan 100% automatic, the drone needs to have the ability to sense other objects, and it needs to sense them from a distance far enough to make a change in its flight coarse. We propose two types of sensors that could enable this feature in the drone: Proximity sensors, and Sonar sensors. We need to test these two types of sensors to determine which is more reliable.
2. **Package Delivery Mechanism:** The main goal of this project was to build the framework of the autonomous drone delivery. We have achieved the goal of flying the drone from point A to point B automatically and safely. What comes next is the development of package delivery mechanism. There are several solutions proposed for this feature. This feature can be broken down to two factors:
   1. Holding the Package: The way the drone will carry the object is yet to be determined. Some methods ensure the safety of the package so that it won’t fall from the drone, such as using a container box and putting the package inside of it. This method will also protect the package from being scratches if it was held directly by a claw, and also the damage caused by weather. The disadvantage of this method is it will increase wind resistance which will lead to more battery power consumption hence less distance to cover.
   2. Identifying the Correct Customer: The drone should have the ability of delivering the package personally to the person who placed the order. This feature is very important if the package was a legal document or medication. In these cases, having the drone to just land in the garden or the backyard then dropping the package is not an option. We propose two methods at the moment to apply this feature. Facial recognition, and lock code. Facial recognition is helpful in identifying the right person, and decreases the chances of delivering the package to the wrong person. A lock code on the package could be an extra security checkpoint, where a lock code could be given to the customer after the order is placed, so only him/her can open the box and take the package out. Electronic locks are the best in achieving this, since they can be used quickly.
3. **Battery Power:** Improving the battery power has two obvious advantages: Extending flight time and raising the weight limit of the package.
4. **Operation Algorithm Development:** All the previous features need addition of code on the software end. The sensors must have a driver that can process the readings of the sensors as an input, and changes the flight plan accordingly, which means sending flight commands as an output. The package delivery mechanism should also have a driver that could tell the drone whether the person standing in front of it is the right person or not, then either release the package or raise altitude to prevent the person from stealing the package.

**References**

Android GPS

<http://www.vogella.com/tutorials/AndroidLocationAPI/article.html#locationapi>

Troubleshooting the arducopter:

<http://code.google.com/p/arducopter/wiki/AC2_Troubleshooting>

Understanding the PIXHAWK:

<http://copter.ardupilot.com/wiki/common-pixhawk-overview/>

Raw Code for Arducopter

<http://dev.ardupilot.com/>

MAVProxy

<http://tridge.github.io/MAVProxy/>

Configuring Python

<http://www.anthonydebarros.com/2011/10/15/setting-up-python-in-windows-7/>

Arducopter Simulator

<http://dev.ardupilot.com/wiki/setting-up-sitl-on-linux/>

API for Google’s Directions within Google Maps

<https://developers.google.com/maps/documentation/directions/>

Configuring the APM level pre-flight.

<https://code.google.com/p/arducopter/wiki/AC2_Flying>

Socket connection

<https://docs.python.org/release/2.5.2/lib/socket-example.html>

DroneApi

http://dronekit.io/

Iris and parts found at:

<http://www.adafruit.com>

<http://www.3drobotics.com>