Autonomous Drone Flights in Response to System Events

A Thesis Presented to

The Faculty of the Computer Science Program

California State University Channel Islands

In (Partial) Fulfillment

of the Requirements for the Degree

Masters of Science in Computer Science

by

Michael James Kaiser

May 2017

APPROVED FOR THE COMPUTER SCIENCE PROGRAM

5/4/2017

Advisor: Dr. Andrzej Bieszczad Date

5/4/17

Dr. Jason Isaacs Date

5/4/17

Dr. Radik Gradinarski Date

APPROVED FOR THE UNIVERSITY

5-4-17

Dr. Gary A. Berg Date

Autonomous Drone Flight in Response to System Events

by

Michael James Kaiser

Computer Science Program

California State University Channel Islands

Abstract

Unmanned Aerial Vehicle (UAV) are a booming subject of research as their impact on our lives increase as they become smaller and cheaper to produce. Recent advances in technology and theory in this field is leading to their ubiquitous use in everyday life for recreational, search and rescue, economic, political, and scientific applications. The CI Rainbow project, which places a wireless sensor network connected to the cloud on Santa Rosa Island, can benefit greatly from these advances. The Parrot AR Drone 2.0 has hit the price point of being an effective tool for exploring the use of drones for autonomous flights in response to a system generated event, and can be a valuable tool for increasing the depth and quality of research possible by the CI Rainbow project.

Integration of drones into the sensor-net adds the capability of autonomously surveilling the environment of a targeted location on the Santa Rosa Island. The ability to respond to a system event that may indicate a fire, flood, or the monitoring and tracking of a difficult to follow animal is an obvious boon to project's goal of studying the island. This thesis paper will break down in detail how an unmanned aircraft system works, how UAVs have been integrated into the CI Rainbow project, and cover in detail the issues that have been run into integrating this technology, provide a look at the recent research into drone capabilities which allow it to overcome some limitations imposed by the environment. Use of the open source Paparazzi autopilot software with the AR Drone 2.0 to implement such a drone will be discussed, and an implementation of some of these technologies will be provided to demonstrate the feasibility of such a project.

Acknowledgements

The author would like to thank the thesis advisor and program coordinator Dr. Andrzej Bieszczad for the inspiration for this project, and constant, enthusiastic help towards making it happen. The author would also like to thank Dr. Jason Issacs, and Dr. Radik Gradinarski for being a part of the thesis committee, and taking the time to review this research. This research would not have been possible without the many hours of online support from the volunteer developers who work on the open source Paparazzi UAV project. Lastly, the author would like to thank Kevin Scrivnor for his assistance with the setup and integration of this research with his research.

TABLE OF CONTENT

TABLE OF FIGURES

Chapter 1: Introduction

1.1 Introduction

The world of technology is booming with new advances and applications of technology that has never been imagined before, providing the tools to improve all aspects of our lives. Computer technology has especially benefitted from new advances that have increased the power and miniaturized the processors of today, enabling the expanding field of the internet of things to take on our desire to stay connected and informed about the world around us. Many fields of research have begun leveraging new technology to increase the quality and quantity of research that can be done, by reducing the need for manually measuring and sensing the environment. One example of this is the CI Rainbow research project, which is the implementation of a cloud connected wireless sensor network, database, and web interface with the goal of being deployed on Santa Rosa Island for the benefit of researchers and educators interested in studying the Channel Islands. This thesis is a report on the research and implementation of the capability for the CI Rainbow sensor network to use unmanned aerial vehicles response to system events to remotely and autonomously sense the dynamic environment of Santa Rosa Island.

The main contribution of this work are the details, requirements, and implementation of autonomous drone flights in response to system events. Autonomous drone flight brings remote sensing to new heights by harnessing small embedded systems to be able to autonomously perform tasks which were not possible before, allowing for research into areas that have been difficult for humans to do. The application of this technology can revolutionize fields of research, sectors of the economy, and improve our lives directly by replacing humans in various difficult tasks, that were once only able to be done manually. In the CI Rainbow project, this technology adds the ability for the network to investigate any interesting event from a literal bird's eye view of the location in which it is happening. This data, can be used to give researches of the island a different perspective, and sense the environment from locations which are otherwise impossible without the ability of flight.

Applications of this technology are not limited to Santa Rosa Island, nor just the events which can happen on this island. Instead of having people observe the skies for smoke, this technology can be used to reduce the response time to catastrophic events in the area by alerting first responders by using data from wireless sensor networks and video captured by drone. Missing persons in the wilderness can be searched for efficiently using several drones rather than an army of people to scan the terrain, which can often be dangerous for those involved. Oil pipelines can be scanned for spills autonomously and efficiently instead of needing human inspection along miles and miles of pipeline. Temperatures can be recorded from altitude without requiring an expensive manned aircraft flight. The possibilities are endless, and therefore this technology is deeply important to develop.

1.2 System Architecture

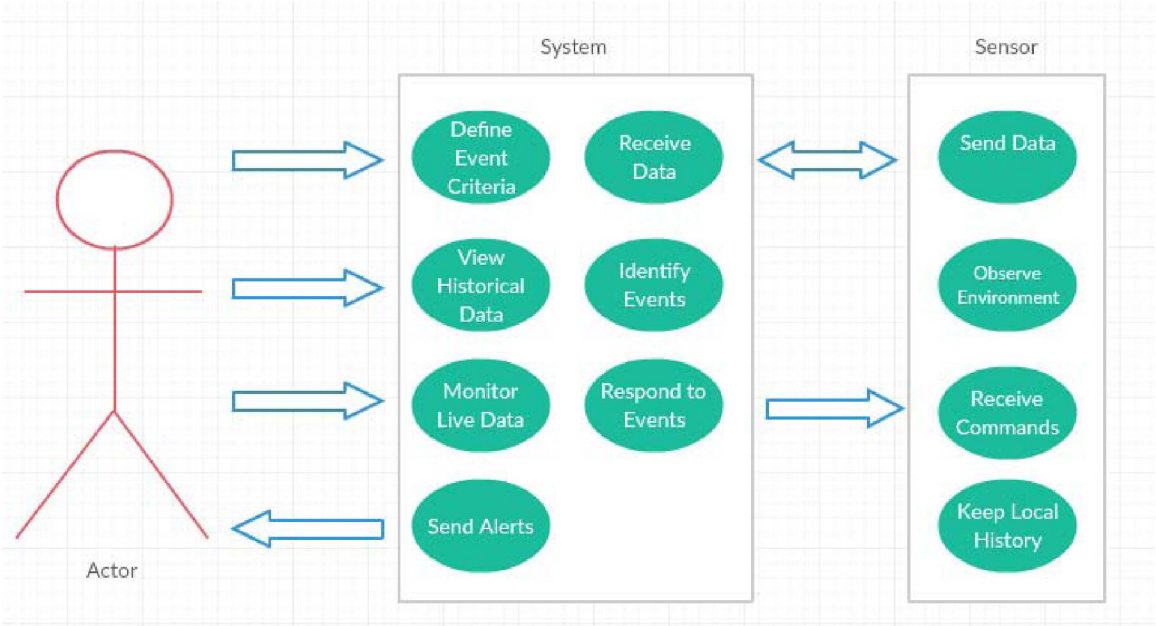

Figure 1. Use Case Diagram

The research that is reported in this document aims to solve a logistical problem with surveilling an area. Typically, when there is a need to observe a place, location, person, or otherwise, a person must be there at the right time to witness an event, which represents an expenditure of resources to facilitate the person's presence. This quickly becomes expensive when events are distributed over a large geographical area that needs observation, and when events occur spontaneously. A private eye or police detective asked about his or her work will often describe the fact that surveillance is difficult and monotonous, whether it is watching a person or a location. The solution for this problem is to automate it, and use the proliferation of cheap electronics to do so.

For any proposed solution to solve the problem at hand, it must be more economically viable than sending people to do the same job, and it must be able to do a set of tasks required of surveilling an area that a person can do. In Figure 1, the general use case of this platform is shown, which is to collect data from sensors and provide the tools for doing so. The sensor in Figure 1 is interchangeable with a drone, as the drone is simply a sensor with locomotion. To elaborate, in a wireless sensor network the sensors are statically placed, and remain within the network, but in this application, the sensor is being placed onto a drone to grant it the freedom to move around, both inside and outside of the wireless network.

From the use case diagram follows a list of functional requirements that describes the system that can facilitate these use cases.

1.3 Functional Requirements

To accomplish the main goal of providing autonomous flight in response to system events, several requirements are imposed upon the system, and the drone. These requirements form the basis for determining whether the project is a success, and is usable, or if the project requires better hardware and technology. These simple requirements will provide for a usable system that can easily be cheaper and more productive than employing a person to do these same tasks. If these requirements are met, then it is reasonable to assert that this application of the technology is viable, effective, and should be applied in situations where it can be used.

1.3.1 Drone Functional Requirements

The drone must have several capabilities and features to be useful for the system, which can be encapsulated by several requirements.

1.3.1.1 Navigation

For the drone to be able to navigate, it must locate itself within several meters of accuracy, and provide an interface for commanding it to those locations. This will be accomplished by using GPS technology, by attaching a GPS module to the drone that allows for the onboard computer to be able to determine its location in space, correct for error, and move to where it needs to go.

1.3.1.2 Communication

The drone needs to be able to communicate with the wireless sensor network, which implies the existence of a wireless transceiver on the drone. This enables the receiving of a command before a flight, and the transfer of data from the drone after a flight to be uploaded to the database. Additionally, it must also function outside of the communication range of the wireless network to capture data away from the static sensors, and to ensure reliability should the wireless signals are disrupted by interference. This alleviates the need to cover the entire island with wireless signal which would be unrealistic. Additionally, there may be negative side effects, both on the environment, and on project costs.

1.3.1.3 Remote Sensing

The reason for the drone's existence is to capture data from the environment in a way that the static sensors cannot, so one of the fundamental requirements of this drone is to be able to capture data using attached sensors during flight. This is accomplished with a camera connected to the onboard computer, allows for audio and video capture of the drone's environment while it is flying.

1.3.1.4 Autonomy

The goal of this project is to eliminate the need for a human pilot and constant observation, so the drone must be autonomous. Autonomous flight, which will be defined more concretely in chapter 2, will consist of being able to navigate to and from a start location to a target location, taking an efficient path, and self-correcting for deviations from the flight path.

1.3.2 System Functional Requirements

The system, or backend server, has a different set of requirements, which are implemented by the CI Rainbow Project, but are critical for the operation of the drone within this project.

1.3.2.1 Data Management

First, the system must receive data from sensor nodes installed into the wireless sensor network, and analyze the data, in order to generate events. The data must be monitored at some interval to be useful for evaluation by the system to generate events, which requires the use of a database to store and retrieve historical data for evaluation.

1.3.2.2 User Interface

The system must provide a platform for the user to configure the server, send and receive data, and manage events.

1.3.2.3 Static Nodes

The system must provide static sensor nodes for generating events. These nodes are to be connected by way of wireless sensor network, and report to the server.

1.3.2.4 Events

The definition, criteria for triggering, and response for events must be able to be defined by the user, stored by the system, and polled for determining if the event has occurred. Once an event has occurred, the system must execute the appropriate defined response for that event in a timely manner. Additionally, one of the actions triggered by the event, must be able to cause drone flight, and handle the retrieval of the data upon return of the drone from its flight.

1.4 AR Drone 2.0

The AR Drone 2.0 developed by Parrot, is the unmanned aircraft platform on which this research was completed on. It has several hardware sensors, components and features which are necessary for the development and application of useful autonomous flight:

- HD Camera (720p resolution, 30 frames per second)
- Video storage on the device
- Cheap, light, and fully reparable structure
- 1GHz 32 bit ARM Cortex A8 processor
- Linux 2.6.32 with Busybox
- Wi-Fi b,g,n
- Three axis gyroscope
- Three axis accelerometer
- Three axis magnetometer
- Pressure sensor
- Ultrasound sensors for ground altitude measurement
- 60 fps vertical QVGA camera for ground speed measurement

Most importantly, this drone can have its firmware overwritten, such that autopilot software can be side-loaded onto the board, and executed instead of the factory software. This is where most of the work for this research went into, which is where the autonomous flight takes place.

The autonomous flight software has the ability receive GPS coordinates of waypoints, and follow them to a destination, and follow them back to the start location. At the destination, a flight routine, such a loop, can be performed to get an aerial video of the environment in that location, which will be reported back to the cloud that powers the CI Rainbow sensor net.

Figure 2. AR Drone 2.0 in Protective Shell

1.5 Remaining Chapters

The second chapter analyzes the fields of wireless sensor networks, unmanned aircraft, and autonomous flights.

In the third chapter, the technical details of the research are discussed, detailing in depth what the problem is, and how each subsystem of the solution working together will solve the problem of providing autonomous drone flights in response to system events on Santa Rosa Island.

The fourth chapter discusses the experiments to be performed to evaluate whether the system works, by looking at the functional requirements of the system, and distilling them into incremental tests which can be performed to ensure a safe drone flight when everything is implemented.

In the fifth chapter, the data from the experiments presented in chapter four are presented, analyzed, and discussed.

The sixth chapter summarizes the conclusions reached and evaluates what is possible with the current status of this project.

The seventh and final chapter will include a summary of the research's results, and will look forward to what is possible with this project if future work is done to improve and expand it.

1.6 Key Terms

- **Autopilot**: a system which takes the inputs of various sensors and maintains the trajectory of a vehicle without human interaction.

- **Autonomy:** Self-sufficient, and self-directed; without human interaction.

- **Autonomous Flight:** the intelligent navigation, route planning, and deployment of an aircraft based on self-governing rules and commands programmed into the computer system of the aircraft, usually with some payload to accomplish a goal.

- **Inertial Measurement Unit (IMU):** The embedded device composed of an accelerometer, gyroscope, and magnetometer that measures and reports the specific force, angular rate, and magnetic field surrounding the device.

- **Ivy:** The text-based publish-subscribe bus used to link elements of the ground control station to communicate with the unmanned aircraft.

- **Paparazzi UAV:** open source hardware and software framework that provides autopilot, ground control station software, configuration, setup, and addition tools for unmanned aircraft.

- **Unmanned Aircraft System (UAS):** The system, including unmanned aerial vehicle, communication interface, and ground control system, which works together to provide for autonomous flights.

- **Unmanned Aerial Vehicle, UAV, Drone**: synonyms for aircraft which do not have a human pilot on board.

- **Waypoint:** a reference point with a latitude, longitude, and altitude for the purposes of navigation.

- **Wireless Sensor Network (WSN)**: A distributed network of computerized sensors for measuring the environment which cooperatively pass data to a central location, or database.

Chapter 2: Field Overview

2.1 Wireless Sensor Network

Wireless Sensor Networks (WSN) are a series of nodes, or autonomous sensors, which communicate their data wirelessly through each other to a central data store, or gateway sensor node, which manages receiving and distributing data outside of the network. These networks have a wide range of applications, including the autonomous monitoring, and reporting of the state of an environment, which is the central goal of the CI Rainbow project.

2.1.1 Nodes

Nodes are small embedded systems with the ability to observe and report the state of the environment. In the WSN, wireless communication is bi-directional, allowing for both the reporting of sensor data, and the retrieval of commands or updates from the other nodes and the remote server. Wi-Fi technology has become less expensive, and requires less power than its alternatives, which has resulted in an increase of its use in wireless sensor networks. For this application, Wi-Fi technology meets the requirements as a communication protocol and it is supported on generic embedded system platforms such as the Raspberry Pi.

The node itself in the CI Rainbow project will encapsulate one or more sensors that detect the state of the environment such as wind speed and direction, temperature, humidity, soil moisture, water quality and pH, sounds, video, and more. This data is what is interesting to researchers and what is sent from the nodes to the main server to be stored and processed.

2.1.2 Central Database Server

The central database server in this application of a WSN is used to collect the data from the other nodes, and provide the data outside of the WSN to researchers or the public internet. Additionally, the server can send data or commands to the nodes, and in the case for this thesis, determine whether an event has taken place, and issue a command to an unmanned aerial vehicle to gather additional data from an event location.

2.1.3 Events

An event is a state that is triggered when a series of conditions are met, enabling an action to be performed in response. Examples of conditions for an event might include when a node unexpectedly stop sending data, when temperature exceeds 30°C, humidity drops to 10%, when the time is 12:00 a.m., or many other possibilities. Combinations of conditions can be taken to form a trigger for an event. When an event is triggered, it can be categorized, and priorities, with certain responses assigned to different categories of events. The response to the event can be anything ranging from notifying by email the administrator of the WSN, to dispatching an unmanned aerial vehicle to collect video of an area. This provides the ability for the WSN to respond to changes in the environment and not require constant human monitoring.

It is also important to consider that the conditions used for triggering an event might include data from real world sensors, and that these sensors are erroneous, or report "fuzzy" values, which may have low precision, or low accuracy. This can be accounted in many ways, from using ranges of values to trigger an event, to using neural networks to detect patterns in reported data to detect anomalies.

2.2 Unmanned Aerial Vehicles

Unmanned aerial vehicles, referred to as drones or UAVs interchangeably, are aircraft which do not have a human pilot on board. These aircraft are piloted by computer systems which are controlled autonomously, or by a remote pilot. Initially developed for military applications, the commercial drone market is currently expanding, and is projected to reach $1.8 billion by 2020, driven by rising demand for their use in high-risk tasks, and the growing prominence of drones-as-a-service [1].

From Figure 3, most of the market segment is military, but there is strong growth in the civil segment [1]. As the popularity of UAVs increases, governments will need to address the legal ramifications of the increased air traffic and economic effects of this technology, which could either hamper or expedite research in this area.

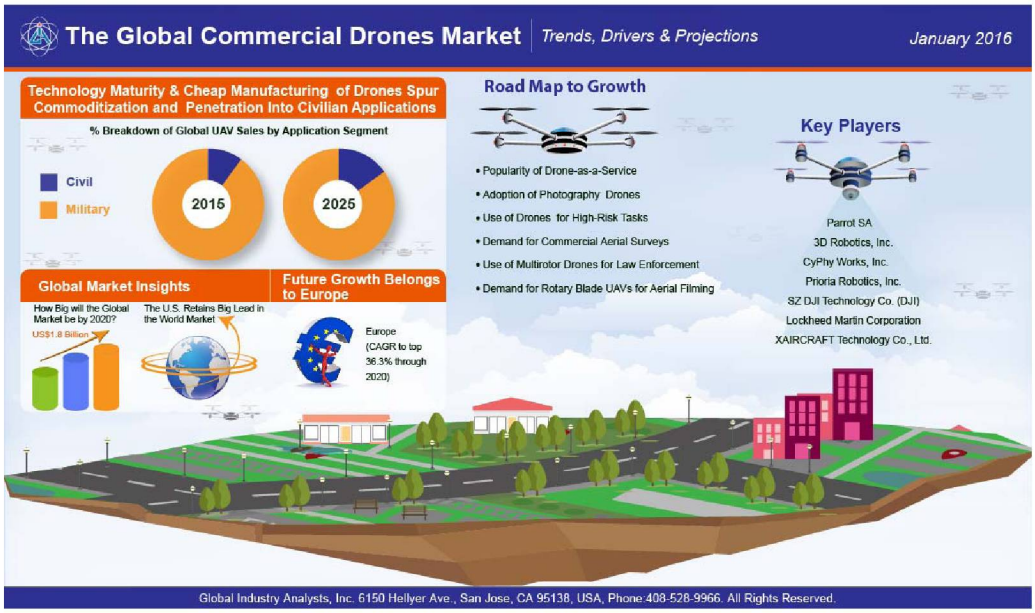

Figure 3. Overview of the UAV market [1]

2.2.1 Anatomy of a Drone

The unmanned aircraft can either be a fixed-wing aircraft, or a rotary-wing aircraft, which are categories of aircraft which encompass conventional airplanes and helicopters respectively. Fixed-wing aircraft require that they maintain a velocity to keep flying, where as a rotary-wing aircraft, or rotocraft for short, can maintain a position in flight by spinning their rotational blades. Additionally, rotocraft can perform vertical takeoff and landing (VTOL) without a runway, which is ideal for congested airspaces, and autonomous flights. Hybrid aircraft do exist, such as the Vertex Hybrid UAV [2], these features are not often found on consumer drones.

The typical hardware that a rotocraft UAV consists of includes an autopilot control board, a source of fuel (battery), antenna for communications, servo motor powered rotational wings, a solid frame to connect these components and a payload deployable during flight. A payload for a UAV are usually sensors to provide for remote sensing, such as a camera and audio recorder, and a hard drive for data storage, such that the flight can accomplish a goal. UAVs are deployed as one part of an unmanned aircraft system (UAS) which includes not only the drone, but the ground control station (GCS) and the data links between them.

2.2.2 Applications of UAVs

Drones are used in situations where manned flights present a challenge. Although intended initially for military applications, UAVs generally perform missions characterized by the three Ds: dull, dirty, and dangerous [3]. This still holds true today, where the applications of drones solve any of those three issues with manned flights, but unlike in the 90s where drones were as expensive as full-sized airplanes, consumer sized drones also can reduce costs, opens the door for business applications, for example, to increase the overall safety and efficiency of the transportation system [4].

As the price of UAV technology has quickly fallen over time, the accessibility of this technology has risen, giving rise to numerous, unique and unexpected applications. Some examples of novel applications of this technology include measuring atmospheric temperatures in the Antarctic using drones to perform the measurement at different altitudes by the Cooperative Institute for Research in Environmental Sciences [5], or an ambulance drone in development by Delft University of Technology to deliver lifesaving technologies in response to emergencies [6]. There are far more well-documented use cases of drone technology that can be listed here, with new uses being invented frequently.

2.3 Autonomous Flight

Autonomous flight is the ultimate application of integrating computers with a vehicle, and it has taken years of innovation to get to the point where this is possible. The complete control of a vehicle by a computer affords several advantages over human control in ideal circumstances, such as the ability to immediately correct for errors in direction or speed on a sub-millisecond level, and to be robust against human error in judgement or perception. The technology is not yet at the point where it is perfected, so it is limited by the quality of the sensors used, and the quality of the models used for calculating adjustments to the control of the vehicle, but these limitations only makes this field of research more interesting and rewarding to study.

There are four key components necessary for autonomous flight in a drone: state estimation, control, mapping, and planning [7]. The first two are primarily the focus of what is referred to as autopilot, and the last two are functions of the program that provides autonomy to the drone, as will be discussed in the rest of this chapter.

2.3.1 Autopilot

The autopilot control board, that resides onboard a drone, is a system which takes the inputs of various sensors, including global positioning satellite receivers (GPS), inertial measurement units (IMUs), altimeters, magnetometers, and feedback from the servo motor encoders to control the hardware by commanding a level of power output to be sent to the servo motors to maintain the trajectory of flight. This system is responsible for ensuring the drone maintains flight where it has been commanded to fly, but it is not responsible for making the decisions of where to fly. Autopilot is a prerequisite for autonomous flight, because without it, the autonomous flight system would be unable to control the aircraft in a dynamic environment. Mathematic properties of the system behind autopilot are responsible for correcting for errors in flight, which will make the aircraft resilient to wind or other sources of error that occur mid-flight.

The autopilot feature of a drone is responsible for estimating the current velocity and position of the drone at an instance of time, and computing the error from where it needs to be, and commands each of the rotors on the drone to spin at a specific speed to correct for the error. The estimation of the current state is mostly done by reading data from the onboard sensors, and the calculation of the error is done at run time, using math equations based in control theory to supply corrective actions for the drone to maintain the heading.

Simply put, there are forces being applied to the drone, such as air pressure, gravity, that divert the drone from its intended location constantly, and the autopilot seeks to estimate and correct for those forces in real time.

2.3.2 Control Theory

Flight is a constant war against gravity, and a drone is a device that can be used in this fight, as it is equipped with tools which can push air below it to maintain some distance from the ground. For this to be useful as an autonomous drone in response to system events, it must maintain distance over the ground over a period of time in a controlled manner to allow the use of the onboard camera, amongst other things. To control itself, autopilot is needed to maintain the tilt, roll, pitch and altitude of the aircraft to maintain trajectory. Autopilots can be implemented in several different ways, but the most common way this is implemented is through a mechanism derived in control theory called a proportional-integral-derivative control loop feedback mechanism (PID control).

The PID control loop is one of the most commonly used feedback control designs because it is a closed loop system which is simple to implement and intuitive to operate. As of 2006, "it is estimated that over 90% of control loops employ PID control" [8]. This type of design is used to maintain flight for the autopilot by minimizing the error, which is the difference between a desired set point and a measurable variable. By minimizing this error, commands to the hardware will provide for the self-correcting behavior of the autopilot. In a tangible sense, this would give the behavior such that if a hovering drone was pushed, that it would return to the position it was hovering at automatically.

Each term of the three terms of the PID controller has a gain, or level of impact on the control of the system which can be independently tuned with respect of each other. A stable system is the result of a well-tuned system, as the terms balance each other out, as they often provide competing signals in response to the error signal. The proportional term is simply the error at a given instance of time, which if left to its own devices, will attempt to correct for the error as fast as possible. The derivative term is based on the change of error at a given instance, such that as the proportional term might drive the system to correct for the error as fast as possible, the derivative term will temper that input as the error reduces. The derivative term also adjusts for sudden changes in error, which are frequently observed in a real-world system. The integral term corrects for steady state errors, or sources of error that are introduced as time approaches infinity, by taking an integral of the error. These three terms, when tuned correctly to the system, give a system stability, when implemented as the controller to the inputs of the system.

Optimizations, or tuning, can be done to PID controls which help address some of these issues, and sometimes are a product of trial and error, but these are usually implementation specific. Tuning can be done heuristically, because there are only three gains to vary.

2.3.3 Autonomy

With a drone that has a functioning autopilot, an autonomous system can be built on top of it. The dictionary defines autonomy as the "freedom to determine one's own actions, behaviour, etc" [9], which emphasizes the ability of the self to make decisions, rather than being told what to do. To better clarify what autonomy is, and to clear up the common misconception that autopilot and autonomous flight are the same, a simple analogy can be used. Autopilot is analogous to the human body's list of autonomic systems, like the cardiovascular system, which maintain homeostasis and life; staying alive is handled in the background. Autonomous flight is analogous to conscious thought, a foreground task, which is being carried out in the mind without regard to the background task, such as beating a heart or breathing.

The autopilot control system in the background provides the framework under which autonomous flight can take place in the foreground, without having to consider the details of maintaining the flight. Autonomous flight is the intelligent navigation, route planning, and deployment of the payload handled based on self-governing rules and commands programmed into the UAV. For any robot, the defining feature of autonomy is the distinct lack of human interaction or micro-management to make decisions while operational.

2.3.3 Navigation

An autonomous drone is free to choose its actions, but its destination and its path to arrive is driven by a function of its initial position, its target position, and knowledge of the environment. The theoretical path planning techniques assume complete knowledge of the environment, but for a drone operating outdoors in a dynamic environment, this is impossible. Nevertheless, using some knowledge of the outside environment in the form of waypoints, it has been empirically shown that the use of waypoints and GPS are powerful tools for outdoor mobile navigation, with multiple projects finding success leveraging this [10].

Chapter 3: Technical details of the work

3.1 Overview

The task of implementing a system that provides for autonomous drone flights in response to events can be tackled with a divide and conquer strategy, as all segments of the project can be handled without the others, as most elements of the project can stand alone, and are loosely connected. The project can be separated into two major areas of focus, the unmanned aircraft system, and the backend system that work together to accomplish the goal of providing autonomous remote sensing data to researches from Santa Rosa Island.

3.2 Sensor Nodes

The sensor nodes are simply small weatherproofed units that combine a sensor and a Raspberry Pi processor board. The Raspberry Pi Zero is a small, $5 embedded computer board developed to aid in the teaching of computer science and engineering, but is used in the CI Rainbow project as the foundation for the nodes of the wireless sensor network. The board has several features making attractive for building a sensor node on, including onboard micro USB and GPIO, and a low power requirement of a 5V power source, which is maintainable using solar power and a small battery. The board itself runs an ARMv6 processor with Raspbian Linux as an operating system, and custom software and configuration based on the type of sensor connected to the node. These devices can be networked together using a USB Wi-Fi adapter, enabling them to communicate with the wireless sensor network. The backend of system has been designed to generically handle various types of sensor payloads, such that several different types of sensors are possible to hook into the system. Initially, the system intends to have sensor nodes with a selection of thermometers, hygrometers (humidity), anemometers, microphones, cameras, and soil moisture sensors to generate data about the environment the sensor is placed in.

3.2.1 Sensors

This is a summary of the sensors initially used with the project and their capabilities, as the sensors are not the focus of this thesis, but their data can be used for the basis of defining events and logic within the system to trigger flights, or prohibit drone flights. Additional sensor types are possible to connect to the Raspberry Pi Zero board.

- A DS18B20 programmable resolution 1-wire digital thermometer can be attached to the Pi Zero's GPIO which records a range of temperature readings between -55°C to 125°C. The sensor needs a waterproof casing.
- A generic anemometer sensor found on the adafruit website can be connected measuring from 0 m/s wind to 32.4m/s wind. [10]

- Generic USB microphone sensor. Information about the actual model used isn't currently available.
- Generic USB camera. Information about the actual model used isn't currently available.

3.2.2 Communication of Data

Each node must be connected to the wireless sensor network, which is relying on Wi-Fi as the medium. Network nodes use the AirMAX protocol. Wi-Fi itself is range-limited, but to extend the range, each node can communicate with another node to pass its data from node to node until it reaches the destination. If there is always a path from each node to the main database, the network is preserved. There is the ability to poll the sensors and upload data at configurable and independent rates for each node. The data packets sent are encapsulated as a raw JSON format objects to ensure generic handling of any kind of sensor.

3.3 Database Server

The database server in the wireless sensor network is responsible for aggregating the data reported from the sensors and provide an interface for students and researchers to retrieve and analyze this data. The database server is implemented using a MySQL database running with a Django generated front end. This provides a simple, open source and modular platform with the goal of making development and maintenance as easy as possible, considering that this will be worked on by university students instead of paid engineers. A Representational State Transfer (REST) architecture is used for communication between sensor nodes and the web application.

3.4 Events

Events in this system are identifiable occurrences which have been defined by the users of the system as being significant or desired to be cataloged. These events are defined by the users of the system by specifying several sufficient and necessary conditions which must be true for the event to be considered to have happened, or triggered. Once an event has occurred, it can be logged by the system and handled appropriately. In a more tangible sense for the CI Rainbow project, events are something that has happened on the Santa Rosa Island that should be stored into the database.

From an application querying data and data from the database, the data collected can be used to determine the existence of an event. These events can be stored back into the database for the researches, and can be fed into a system which can process these events based on additional criteria. What this does for the CI Rainbow project is provide a hook into monitoring the data which can trigger an external process to be called when something of interest happens, such that a human does not have to initiate the process. From this system, further autonomous action can take place, such as sending an autonomous drone to investigate an event which warrants addition investigation.

3.4.1 Definition of Events

Categories of events can be defined based on what conditions define the event as having occurred, which should be a combination of data from the database and real-time data, and other sources of information, such as time, to be considered for determining the type and existence of an event. Events are defined by the user which associates a trigger with an action, where the trigger can be defined using data from one or more sensor nodes. When the sensor data is polled, the Events are queried for any triggers to fire, which will launch a process to execute the defined event, which might be the start of an autonomous flight.

3.4.2 Event Detection

Events objects need to be created when their counterpart takes place in the real world, and this is accomplished by an application which has access to all the required data needed to determine when an event has occurred. This application must read data periodically, and be triggered when new data is available, to give full coverage over the data to determine when an event occurs in a timely manner.

Data used as criteria for the event taking place will be coming from several sources where the quality of data must be accounted for to prevent false positive and false negative events from being detected. In other words, the quality of the definition of the events to be used in the system will have an impact on the ability to detect, and correctly respond to them. Furthermore, this takes place in a wireless sensor network where communication with all nodes all the time is not expected to be available, so looking at historical data to detect events is necessary.

Given a system with predefined events, the application will iterate over each type of event and determine whether their conditions are satisfied for determining that they have occurred. Additionally, checks must be done to ensure that this event is reported only once per unique occurrence, to avoid over flooding the system with unnecessary data.

3.5 AR Drone 2.0

The specifications of the AR Drone 2.0 unmanned aerial vehicle are described in Chapter 1, but for the purposes of this research, additional work on the drone is needed to meet the requirements of the project, considering that this model of drone does not provide for autonomous flight out of the box.

3.5.1 GPS module

To use autopilot software on the drone, a position system is required to be placed on the AR Drone 2.0, and the easiest off the shelf solution for this in an outdoor environment is the use of the Global Position System (GPS) which can be utilized by affixing a u-blox LEA-6H GPS receiver unit to the drone, the device pictured below. This unit was chosen because it is accurate within a few meters, which meets the functional requirement for navigation.

Figure 4. u-blox LEA-6H GPS Module w/Built-in Antenna (2.5m Accuracy)

The connection to the drone is made possible by using the USB port connected to the hardware of the AR Drone 2.0's mainboard, which allows for communication using a USB serial cable. The LEA-6H module is wire connected, so an intermediate TTL to USB Serial converter cable is required to connect the two together, with their soldering required. The wires are connected red to red, black to black, yellow to yellow, and green to orange, as seen in Figure 5.

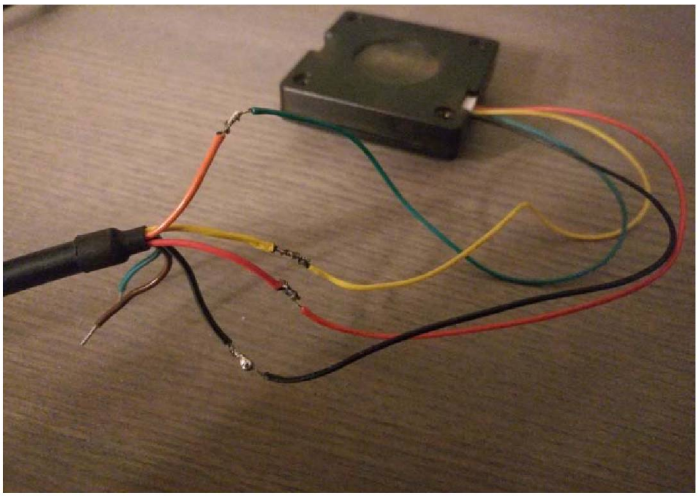

Figure 5. Wire connections with LEA-6H module and serial cable [11].

3.5.2 GPS software

Once the GPS unit is connected, it will appear available to the operating system as a device, but the driver for it does not exist on the board, as it is a minimal Linux build. This is remedied by finding on the internet the CDC-ACM kernel module, and uploading it to the drone through telnet, and inserting it into the kernel using the insmod utility which is available on the board. For use after the initial load, the boot up script on the board can be modified to automatically insert this kernel module on power-up of the system, so this only needs to be done once.

3.6 Paparazzi UAV

Paparazzi UAV is an open source drone hardware and software framework, which provides autopilot, ground control station software, configuration, setup, and testing tools for a multitude of aircraft. Paparazzi was chosen to be used in this project, as it has implementations for most of the tools necessary to build autonomous flight software on top of, such that autonomous flight could be the focus of this research. Importantly, Paparazzi integrates seamlessly with the AR.Drone 2.0, which is the model drone used for this research. To perform autonomous flight with the drone, several steps must be followed to calibrate the sensors and configure the software to work with the specific type of aircraft that is being used.

3.6.1 Configuration

First, the mapping of the yaw, roll, and pitch must be tested against the default configuration by plotting the throttle data while using a hand-held controller to control the drone, and holding the drone in the other hand to confirm the mapping. A Logitech F710 wireless gamepad was used for this purposed, as it was cheap, simple to setup in Linux, and it is wireless. The controller requires a driver to be setup in Linux, and a configuration file for mappings of controller buttons to drone commands be used in Paparazzi. From my experience, most of the keys required swapping of default mappings to be correct, but otherwise worked with the default configuration using the alternate keys.

Next, the magnetometer needs to be calibrated for the location on Earth where the flight is intending to take place, and this is done by taking readings from all different orientations of the drone in relation to the ground, and the Paparazzi data plotting tool provides a graph to ensure most orientations were covered. After confirming the uniformity of the data collection, a Python script is provided by Paparazzi to analyze the data collection and provide constants for the neutral readings and sensitivity to use for the IMU.

3.6.2 Flight Plans

With a configured drone, Paparazzi allows for the implementation of a flight plan which corresponds to actions which the drone can take, and defines the location of where each landmark or waypoint is on the map. These waypoints can be dynamically moved after the drone has the flight plan loaded by sending data over the IVY bus while the drone is connected to Wi-Fi. The flight plan then, is used as a tool to define the number of waypoints to be used, and some pre-defined flight routines to allow for autonomous flight to take place on top of it.

3.7 Navigation

The way in which the drone navigates to and from the target location utilizes the GPS module, Paparazzi, and a small Python program to calculate the shortest path between two points in a graph. The reason this method was chosen was to keep the system as simple as possible, and because it is difficult to implement more intelligent navigation systems with the limited sensors on the AR Drone 2.0. This system requires the preplanning of physical waypoints on Santa Rosa Island and the definition of safe paths between these points. These physical (terrestrial) waypoints will be mapped to logical (programmed) waypoints in the Paparazzi software to command the drone to fly to a location. In this system, there can be an unlimited number of physical waypoints, but Paparazzi has a limitation on the number of logical waypoints at 256, which does not pose a problem.

3.7.1 Shortest Path Generation

The method used to generate this path is a simple Python implementation of Dijkstra's Algorithm, which efficiently iterates over all possible paths to generate the shortest path to the target location [13]. This was chosen as the algorithm to use because it is a well-studied solution to the shortest path problem, provides the optimal shortest path solution which gives the drone the largest range on its limited battery life, and the number of waypoints being used in this research will not cause the runtime of this algorithm to become a problem. For a more general solution where the number of waypoints could be larger, a heuristic solution could easily be substituted for Dijkstra's Algorithm to reduce the computational work at the cost of a longer path, and less effective range of the drone.

3.7.2 Waypoints

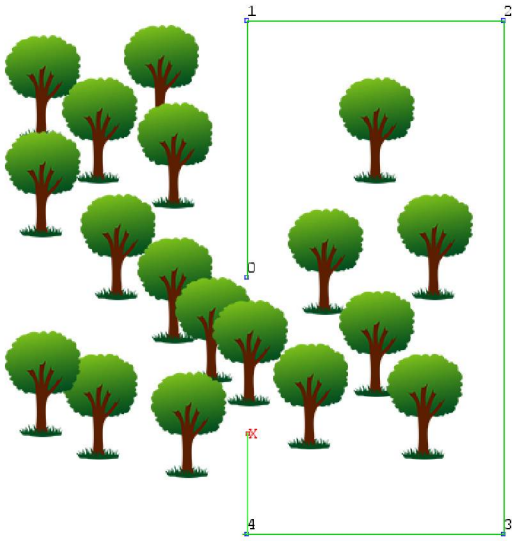

Figure 6. Selection of waypoints to avoid obstacles.

Waypoint based navigation was chosen for its simplicity, ease of implementation under the restraints of both the hardware, and software, and for its ability to be used to avoid obstacles. Being the outdoor locations with specified GPS coordinates, waypoints are selected in combination with defining straight-line paths between them such that known paths around Santa Rosa Island can be safe for the drone. A theoretical example is given in Figure 6, such that placing waypoints strategically will cause the shortest path generating program to output a path that avoids obstacles. If the assertion is made that events will only trigger on top of the defined waypoints, this gives a further guarantee that the drone's flight will be as safe as the input map was defined. There will be no indeterminate distances.

Paparazzi waypoints, which is how the drone navigates, are defined in the flight plan without being bound to physical waypoints until the shortest path is generated, as the logical waypoints are limited in number, and can be moved dynamically. This allows for flexibility in the length of the path, up to the limit of Paparazzi waypoints available. Excess waypoints, given a very short path from the shortest path generating program, are moved on top of the target physical location such that they do not impact the navigation of the drone.

3.8 Autonomous Flights

Performing autonomous flight involves using additional software to communicate with the drone through the services provided the Paparazzi suite, which has been implemented using Python. This software communicates with the physical drone, but is untested with a drone, because rules and regulations prohibit actual flight. The commands will be sent to a simulated drone and its path plotted on the map by Paparazzi instead, which will validate the work done sufficiently for this thesis, but without the benefit of field-testing.

3.8.1 Server to Paparazzi Communication

The server will generate events which are associated with GPS coordinates of the occurrence, which will be the target location of the drone. From that, the software that receives the event from the server will use a pre-defined map of GPS waypoints to find a path to the target location and back. The pre-defined map of GPS waypoints are chosen based on known point to point safe routes, such that a list of them will put the drone as close as possible to the event, which may or may not lie upon one of these waypoints, before commanding the drone to fly directly to it.

Once the shortest path has been found and stored in memory, the Python application will communicate the path using TCP/IP to the computer running the Paparazzi UAV suite. TCP is used here to ensure that all data is sent and error checked as the path is not generated during actual flight. The receiving application must take each waypoint in order and move the Paparazzi waypoints to lie upon the shortest path. This is done using the IVY protocol to Paparazzi, which will then provide these waypoints to the drone while it is still connected. Once all waypoints have been setup, the receiving application will command the drone to take flight, and await its return.

3.8.2 Paparazzi to Drone Communication

The Paparazzi application launches the IVY bus which is an open communication protocol between the drone, and the software, and can be used by anyone on the network. When the path waypoints are received, the IVY bus is used to move each waypoint defined in the flight plan to the coordinates specified by the path. With the current implementation, up to 64 different waypoints can be specified, where if the path is shorter than 64 different waypoints, the remaining waypoints are moved to the target location.

The flight plan defines a path between waypoint 1 and waypoint 64 back to waypoint 1, for a departure and arrival back to the start point. At waypoint 64, the flight plan can use a defined "block" to set the behavior of the drone once it arrives at the target destination. For testing purposes, this is simply defined as commanding the drone to circle the target location for 60 seconds using a 10m radius. The possibilities are endless here, for example, advanced logic can be programmed in C, and called by a routine in the block to perform digital image processing from the camera to track a target.

Chapter 4: Experiments

4.1 Purpose of Experiments

Each subsystem of the project that provides autonomous drone flight in response to system events can be individually verified by testing it independently, which provides a clear path for determining if the project has accomplished the stated goal. It also gets around the legal restrictions against flying the physical drone currently in place at CI, as test involving physically flying the drone can but done in the virtual world. Each test is intended to verify that the requirements of the project are met. If all experiments produce positive results, it provides a clear basis for claims that the drone would work autonomously in the real world in response to system events.

This chapter will discuss the setup for verifying each subsystem of the project, which will be followed by the results and discussion in the next chapter.

4.2 Verification of GPS

For the drone to fulfill its requirement of being able to locate itself within several meters, the GPS unit is tested to be accurate. If the GPS is inaccurate, it is impossible to safely fly the drone. The GPS unit, the u-blox LEA-6H module, can first be tested by connecting it to a PC, and connecting it with the freely available u-center software for configuring the device to ensure it can acquire a fix. This is a simple pass or fail test that will confirm whether the module has been wired to the Serial to USB converter correctly, that the hardware is functional, and that the GPS satellites flying overhead are operational.

Next, the connection to the drone is tested, as this involves several differences over testing on a Windows PC, namely that a Linux driver is needed for the GPS module, and that the USB port from the drone has the capability to provide both the power and datalink necessary to operate. This can be tested by verifying the light indicating a GPS fix blinks when attached to the drone, and that correct data is being received in the Paparazzi software.

After the connection is confirmed, the accuracy of the unit in motion can be tested without flying the drone by placing the drone and laptop in the passenger seat of a car, and driving around in a car, such that a path of the drive is drawn onto the map view in Paparazzi. This path can then be compared to the path drawn on a smartphone using any free run tracking application, and the error in the path can be evaluated to determine if the error is within the specifications of the hardware, and otherwise acceptable.

Alternative to using run tracking software, if driving on simple streets and keeping track of what lane was used would be a great way to know what the actual path the GPS should have plotted. It is sufficient for this project to have a simple visual comparison between the GPS data from the drone and the GPS data from a smart phone, but if an accuracy criterion was specified by this project, specialized tools could be used to compare time and position data quantitatively.

4.3 Verification of Drone

The ability of the drone to orient itself is necessary to be tested. The drone's hardware must be safety checked prior to remote controlled flight, as this will verify that the drone can fly correctly, and prevent damage to the drone if it were to malfunction during an attempted flight and crash. These tests do not require positional verification of the drone, and can be performed indoors. The first thing to test is the sensor data to ensure that it matches what the sensors should be reading, and modifying the stimulus to the sensors to ensure they change as expected. The AR Drone 2.0 has several sensors required to self-regulate its flight. To ensure nothing is malfunctioning or damaged, it must be checked to ensure a safe flight each time. By loading the autopilot software through Paparazzi, connecting to the drones broadcasted wireless network, and opening the Ground Control Station will allow the sensor data to be accessible through Paparazzi on the laptop.

First, the Primary Flight Display (PFD) tab in Paparazzi can be opened to show the drone's current orientation. By moving the drone by hand, it can be verified whether the drone was able to calibrate its orientation to the ground properly. Testing at all angles and ensuring a match between the drone's handheld orientation and the on-screen depiction of the drone's view of the horizon will give confirmation that the gyroscope is functional, as demonstrated in Figure 7. A successful test would be showing that the screen matches the orientation in a timely manner, such that the drone will be able to correctly analyze its state.

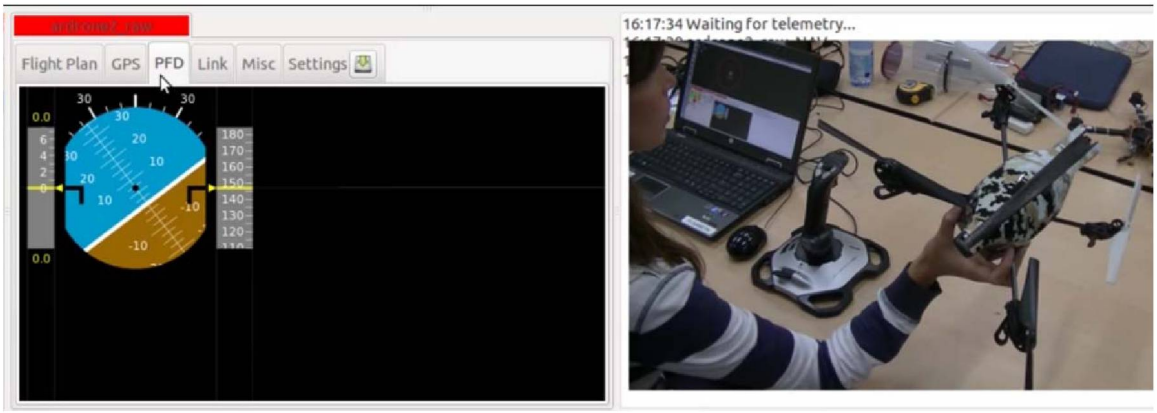

Figure 7. Screen capture of the Paparazzi GCS in the PFD tab. [12]

Next, the drone's flying capability can be verified by using a joystick to manually control the yaw, pitch, roll, and thrust of the aircraft. The joystick's configuration must also be confirmed by using the data plotter within the Paparazzi software to ensure that the joystick's command matches what the correct command for the aircraft should be. The data sent to each motor can be graphed using Paparazzi's plotting tool, and confirm that it matches the motor's behavior in the physical device.

Once the controller is calibrated correctly, the yaw, pitch, and roll of the drone can be tested independently while the drone is being held by hand to prevent it from flying. When working correctly, the propellers opposite of the tilt direction of the joystick will have more thrust. If holding the drone tightly and off the ground, it should be felt in the arm holding the drone which direction the drone is attempting to move to, and this should be compared to the command being sent by the joystick to confirm that they match.

4.4 Verification of Autopilot

The autopilot system of the drone is provided completely by the Paparazzi software, but it is still important to verify that this system is working as expected prior to attempting any autonomous flight, as errors in the autopilot will result in a catastrophic failure. The autopilot software can be tested by entering "altitude mode" in the GCS, or by hotkey on the remote controller, which will use the software to ensure the drone maintains the altitude commanded by the controller.

To test this is functional, moving the drone by hand quickly from the maintained position will cause the motors to thrust harder corresponding to a correction back to the original position when working correctly. To fully test the ability of the autopilot software to maintain a trajectory, a real flight needs to take place, but in lieu of that, the testing of the autonomous flight can apply a wind value to the drone midflight to observe its correction behavior. For the AR Drone 2.0, the autopilot has been specifically developed for this airframe, however, using a different drone system might require additional tuning to ensure that the autopilot works correctly.

4.5 Verification of Shortest Flight Path Generation

The application which accepts the GPS coordinates of the target location for the drone to fly to, and outputs the shortest path from the hard-coded waypoints and edges based on locations on Santa Rosa Island, can be verified to work by testing different target GPS coordinates and visually inspecting to see whether its passed. Several test cases should be run to confirm that it handles the edge conditions, such as testing when the target location is exactly between two waypoints, the target location shares the coordinates of a waypoint, and the target location is outside of locality of the other waypoints. In each test case, the path should be confirmed to be the shortest possible path, as Dijkstra's algorithm is supposed to ensure this.

4.6 Verification of Autonomous Flight

Autonomously flying in response to system events is the defining test for this project to be able to accomplish the overall goal, but because of the restrictions against flying in place at CI at the time of conducting this research, this must be done using the software to simulate flight instead of a real flight. The same test can be performed on a physical drone once the legal restrictions are lifted, which is thanks to the simulation software built into Paparazzi, which can be switched from simulated to real flight seamlessly. The Paparazzi GCS simulation mode sits on the same laptop that would be used to command the drone, and the shortest flight path generator will compute the shortest path to a target GPS location given to it by the event. Verification of success has been done by observing the path that the drone takes on the screen of the ground control station, and ensuring it matches the shortest path generated by the shortest path program, and that it returns at the end of the flight following the same path. To verify that adherence to the path is acceptable in a windy environment, Paparazzi provides a tool to play with the wind value used in the simulation to test that the drone is resisting elements that would cause it to drift from its path.

4.7 Verification of Events

Verification of the event system is to guarantee that events are triggered when their conditions are met, to ensure that a defined action takes place when its trigger is activated, and to test that the drone system will integrate smoothly with the server backend. This will ensure that the requirement that the drone and the CI Rainbow backend work together to trigger flights is satisfied. The simplest way to test this is to manually trigger an event corresponding to a drone flight, and then to confirm that the script to launch the drone has indeed been executed. The main event system, including triggers and actions, have been independently verified by work done by another graduate student who developed the backend server, so that capability is assumed here to function properly.

Chapter 5: Analysis of Results

5.1 Overview

Because this project consists of hardware operating in the real world, moving over time, it is challenging to illustrate some aspects of the experiments working without video demonstration. Screen captures and photographs will be used in this document, with annotations to describe the moving parts in the diagrams as necessary.

5.2 GPS Verification Results

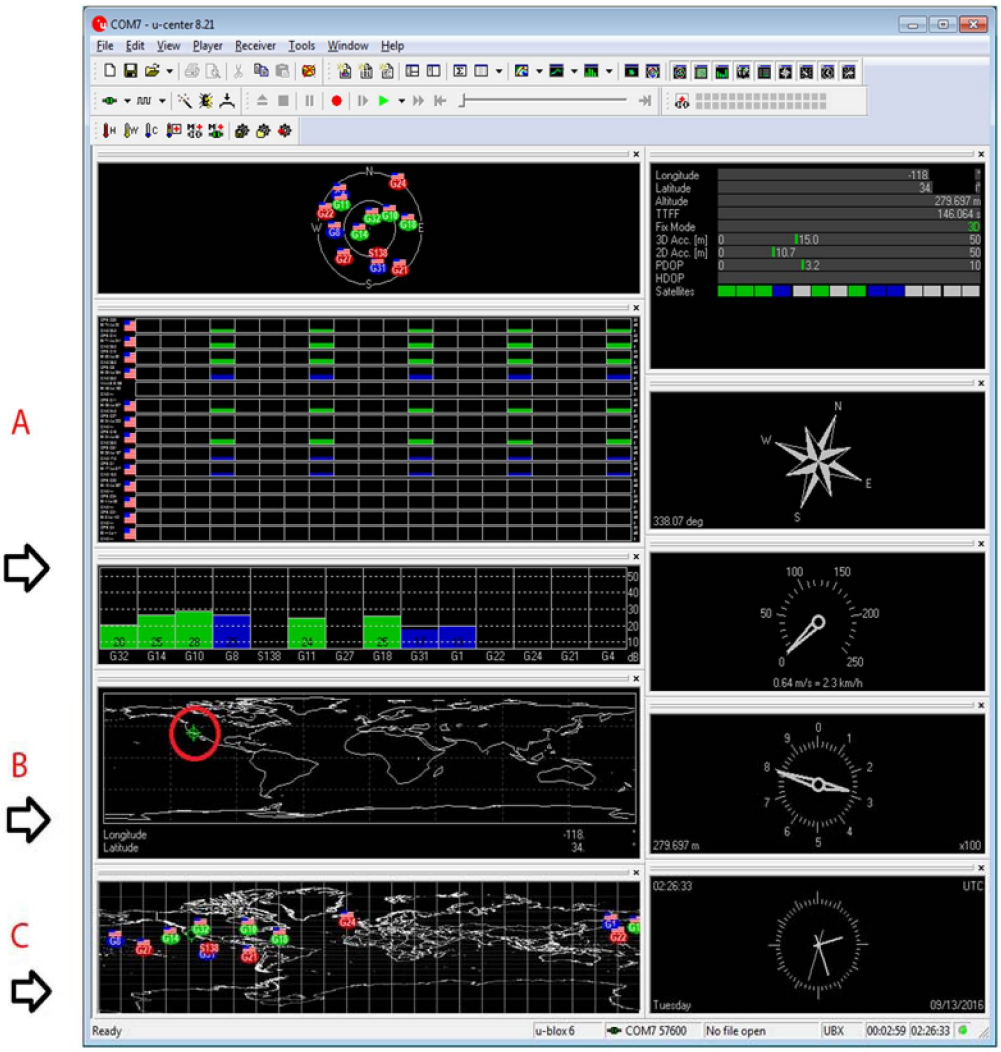

Figure 8. u-center 8.21 display of GPS module with a 3D fix on its location.

Following the experiment described in the previous chapter, the u-blox module was connected to a Windows 7 PC with u-center 8.21 displaying the GPS fix corresponding to the physical location of the GPS module. In Figure 7, the u-center software is shown when a GPS fix is acquired, with several different data displays indicating the information that the module collects. A connection to several GPS satellites is indicated in green and blue bar graphs, and a crosshair drawn on the world map of where the GPS unit indicates that it is located.

In section A of Figure 8, the individual satellites that the module has successfully connected to show their relative strength, and section C plots the satellites general position in the atmosphere. Section B indicates the coordinates and location at which the GPS module is located on the planet, with a crosshair drawn as an indicator.

A drive around a local park with a laptop, the drone with GPS connected, and a smartphone as a baseline, provided the screenshots and pictures to verify that the GPS is tracking properly (Figure 9 and Figure 10).

Comparing the route taken to the mapped path, it is evident that the GPS module is accurately reporting its location from its connection to the drone to the Paparazzi GCS. The GPS path closely matches the path driven, as confirmed by ensuring that it was driven on the right side of the road in Figure 9. In Figure 10, it accurately captured being driven through the parking lot in a zig-zag pattern. The observed accuracy is sufficient for the required tasks of the drone in this application.

Figure 9. GPS Position of the drone capture while driving

Figure 10. Enlarged view of zigzag maneuver captured by GPS plot

5.3 Drone Verification Results

Using the Logitech G710 controller, several keys had to be remapped because the Linux driver differed from what Paparazzi expected each key to be bound to. After remapping the keys, the control of the drone matched the expectation of what the drone would do in all cases in the lens of roll, pitch, yaw, and throttle. Both the physical drone, and the messaging from Paparazzi to the drone were tested to ensure the entire system works as expected. Using the analog sticks on the control, it was easy to confirm the ability to send different strength values for the yaw, pitch, roll, and throttle, as seen in Figure 11.

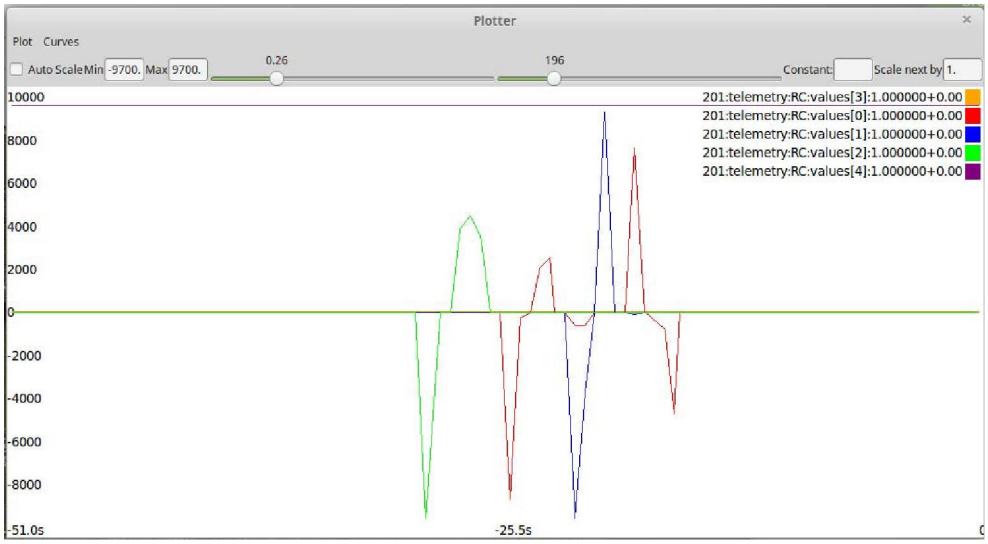

Figure 11. Paparazzi plotter showing the Logitech G710 sending commands to drone

5.4 Autopilot Verification Results

The autopilot test confirmed that when set to a position, the drone will autocorrect against any outside force such as a gust of wind. Resisting change in motion was successful in generating a response when it detected a difference in roll, pitch, yaw, or altitude was detected. The image in Figure 12 below depicts the motor response to differences in orientation of the drone. The graphs on the laptop show the logical commands, yaw, pitch, roll, and throttle, to the drone, and react when the drone's orientation is moved from neutral. To lower the physical pitch which was raised by hand, the software pitch was sharply commanded down, visible in the image to the right, which causes the front two rotors to spin backwards at a high rate.

Figure 12. Commands to the drone are plotted in response to changes in orientation.

To confirm that the orientation, or yaw, pitch, roll and altitude can be maintained, the drone was tested indoors without the use of GPS. While a small amount of positional drift was observed, the other parameters fell within the desired specifications. The positional autopilot is reliant on GPS readings and any GPS readings from indoors are likely reflections of the GPS satellite signals from outside, and are less accurate than a line-of-sight reading for the GPS signal that would be possible outdoors.

To test the longitude and latitude corrections of the autopilot for this project without being able to fly outside, the simulator was used, and covers this testing, which will be discussed in the subsection for autonomous flight.

5.5 Shortest Path Verification Results

The program that generates the shortest path between two waypoints is simple to verify. Typically, only one solution is possible, however, multiple shortest paths will raise the challenge of choice of path. When multiple solutions are possible, if one of the possible solutions is generated, it would be considered a success.

For the verification, several tests were performed to ensure that it can find the only path, it can find the shortest path when there are multiple paths, and that the path selected will reach the waypoint closest to the target. This means, there is a start point, a target location, and a waypoint which resides closest to the target, and the goal of the program is to find the waypoint closest to the target, and then the shortest path to that waypoint.

Choosing the path that finds the waypoint closest to the target is preferred over the absolute shortest path, as it allows the placement of waypoints to avoid known obstacles, and it reduces the distance of travel between a waypoint and a target that may not reside on top of a waypoint. If targets do not reside on waypoints, the terrain will make any path not between two waypoints risky to the drone, as unknown obstacles could make flight difficult. Figure 13 demonstrates how the placement of waypoints in the map can be used to avoid obstacles, and how the program will find the shortest path based on this logic.

In this first example, placement of the waypoints ensures that there is only one possible correct solution, and it is obvious from the picture that the drone would follow this path. This example did not require performing any math to verify the results, and easily demonstrates how the drone will take the shortest path to the closest waypoint before flying to the target, rather than the shortest direct path between the waypoint and the target.

When there are sufficient waypoints near where the target location might be, the drone will follow safe, defined paths, rather than travel along an unknown path. Ideally, the waypoints will be mapped to sensor nodes placed along trails on Santa Rosa Island which can be verified that they are safe to fly over. The subsequent challenge is to choose between multiple valid shortest paths.

In Figure 14, the correctness of the shortest path is tested with a trivial example, which does not require calculations to prove correctness, as shown in the left and middle images. In the third run on the right, the waypoint is equidistant from waypoint 6 and 12, such that 6 was chosen only because it was defined first numerically in the coordinate map. While either path is acceptable in this case, it is not a realistic to come across this situation with real world coordinate systems.

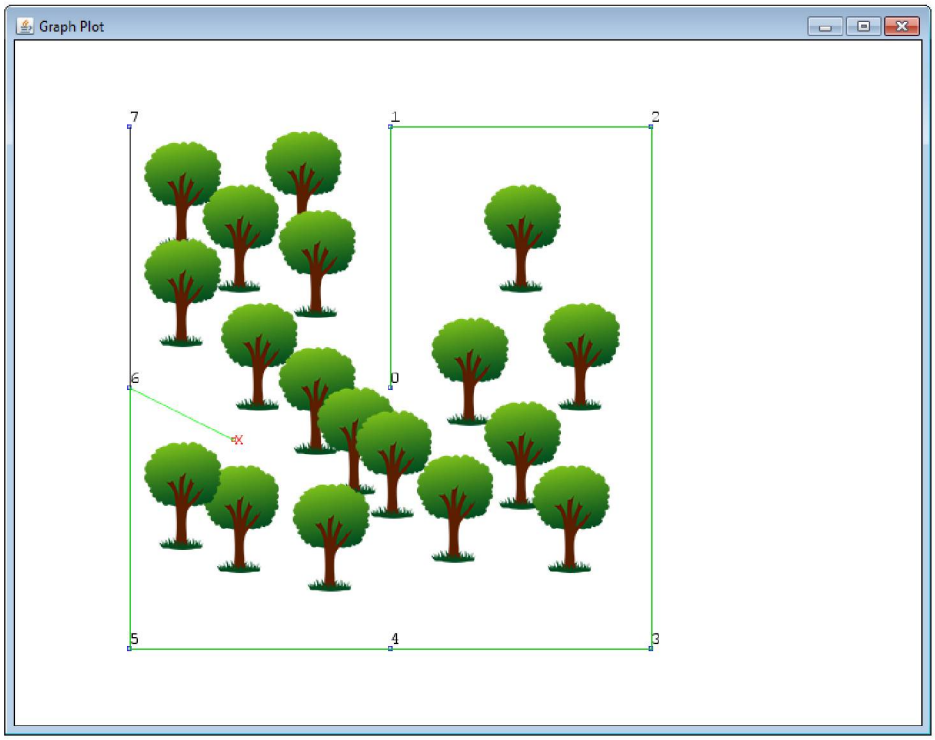

Figure 14. The shortest path is found and highlighted in green

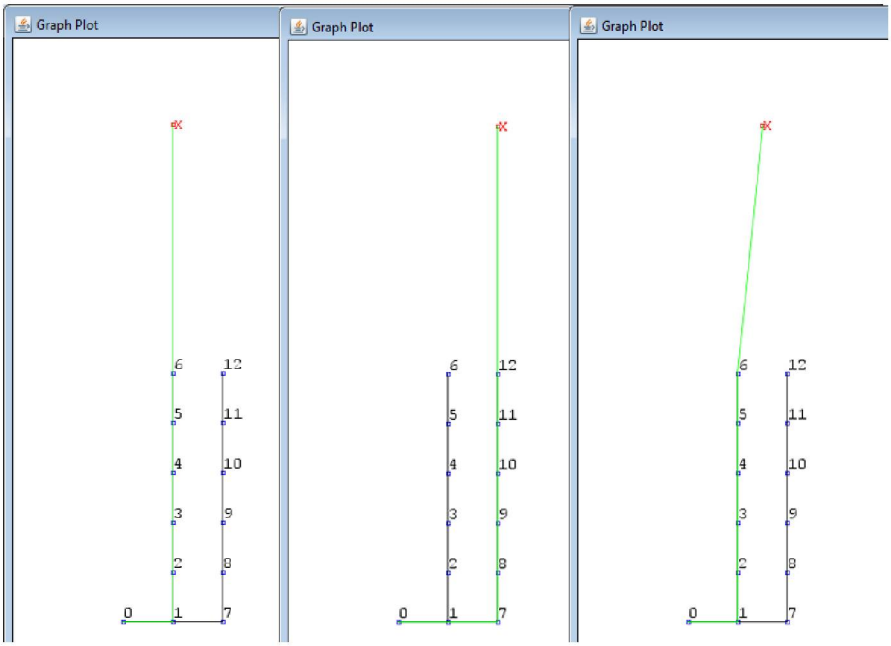

Figure 13. Three tests where the closest waypoint to the target dictates the path

Figure 15. Test using Santa Rosa Island's GPS coordinate space. Shortest path in green.

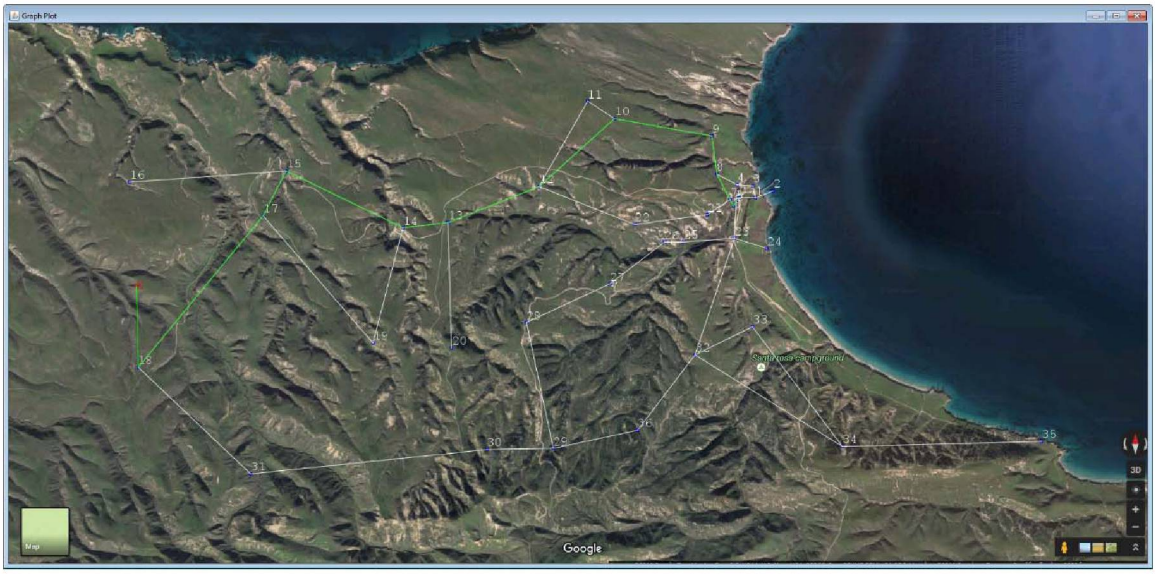

Figure 16. Test where the target location is outside the perimeter of defined waypoints.

The tests in Figures 15 and 16 demonstrates the shortest path program working correctly with coordinates specified in longitude and latitude in a small range of values, from 34.0423343, -120.135776 to 33.972395, -120.002770. Performing floating-point math with a small range of numbers could be troublesome if insufficient precision was used to describe the floating-point numbers.

Note that Figure 15 and 16 were generated by a Java application developed for demonstration purposes to draw the coordinates from the shortest path program, and are not from Paparazzi itself. The plotting program used here is a simple linear map and does not accurately plot the waypoints adjusting for the curvature of the Earth.

The shortest path program does not have a default limit on how far away the target location can be specified from the closest waypoint, but one could be added by setting a limit in longitude, latitude, or both. Additionally, Paparazzi provides GPS gating options in its flight plan to prevent the drone from travelling outside the defined area, which can have the drone fly back the way it came in the event of a bad target location being generated.

5.6 Autonomous Flight Verification Results

Following the experiments in the previous chapter, autonomous flight was tested within the simulation software provided with Paparazzi rather than testing actual drone flight. These tests intend to verify the use cases the autonomous flight system needs to handle. They are based on the shortest path tests to see if the drone correctly follows the path, and returns along that same path. Additionally, the simulation software allows for the specification of variable wind direction and speed during flight simulation, which is used to verify the ability of the drone to self-correct.

First, tests sending the drone to targets around the island were done to ensure that the drone could follow the defined path, and return to the home location without interference. This ensured that when the shortest path was wirelessly sent to Paparazzi, it was received correctly, and moved the logical waypoints appropriately in response to the received waypoints. In Figure 17, the green path is the commanded path, whereas the red path is where the drone has traveled so far. Within the virtual environment, the drone almost perfectly follows the defined path.

When a path has fewer physical waypoints than Paparazzi provides logical waypoints, the excess waypoints are all moved to the last point, such that the drone could simply be commanded to follow waypoints from the first logical waypoint to last logical waypoint. The advantage of this approach is that it utilizes only one flight plan file for every autonomous flight, whereas Paparazzi typically uses one flight plan per flight.

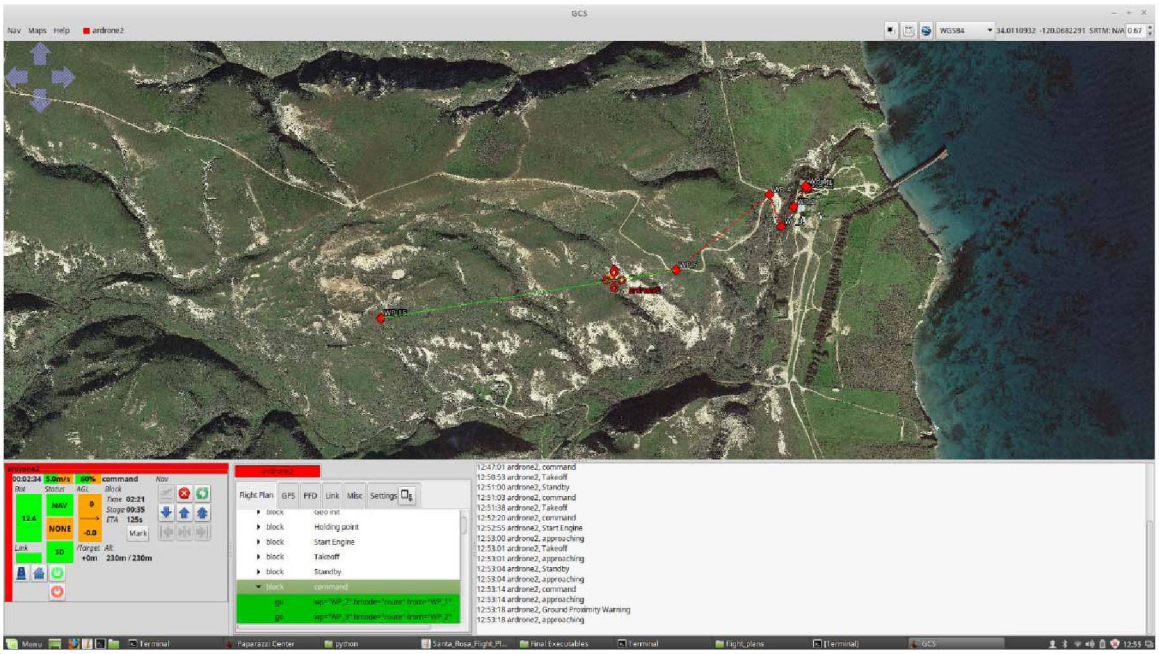

Figure 18. Simulated flight in Paparazzi following shortest path.

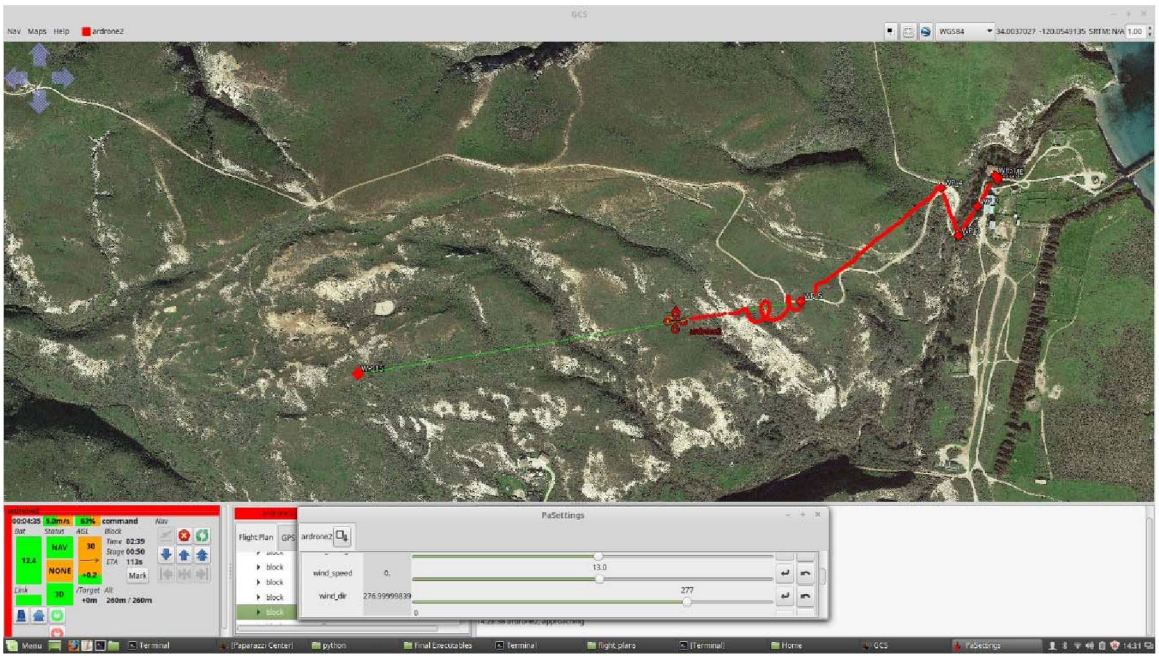

Figure 17. Simulated wind causing deviations and circular corrections back to the path.

In Figure 18, the previous experiment was repeated, but this time, the simulated wind was used to blow the drone from its trajectory. When the wind speed and direction were changed suddenly, a divergence from the path was observed, which subsequently was corrected for by the PID loop. The derivation from the path was correlated to how volatile the change in wind direction and speed was.

Using the Paparazzi wind simulator, values towards the middle of the wind speed slider caused derivation strong enough to cause a circular pattern to appear when the drone attempted correct for the deviations. Although this is a simulation, and it is unknown how closely the simulated wind correlates to wind in the real-world, these circular pattern corrections are possible. If the waypoints are selected to prevent obstacles, it would be a problematic if these deviations from the path were large enough to cause a collision. Values towards the end of the wind speed slider would cause the drone to make an emergency landing, and stop moving, because the deviations from the path grew too large.

There will be occasions where wind or unforeseen obstacles will render a drone inoperable in the real-world. Such eventualities are inevitable. This is performing as best as possible, as it would be unreasonable to expect drone flight to work correctly in extreme wind conditions. To avoid this problem, the backend system could monitor the weather forecast and current wind conditions such that autonomous flights could be prevented when detected at the event layer of the system.

Next, tests were performed to ensure the drone could maneuver around the target area once it has arrived. The flight plan can define multiple subroutines of actions that can be executed, which provides options for how the drone behaves when it reaches the target location before returning.

In Figure 19, the drone is commanded to circle the target for 70 seconds before returning the way it came. The application for this would be to capture a panoramic video of the area when the drone arrives at the target location. This looks acceptable in the simulation, but it cannot be verified whether simply circling the target is an adequate way of capturing all the important information on video. If the event requires capturing footage of an object on the ground, the orientation of the drone and the placement of the camera may miss such an object.

When physical drone flights are possible, the subroutines defined for video capture flight patterns could be developed and tested to ensure that the full range of yaw, roll, and pitch are used to get as much of the area captured on video as possible. Until then, this aspect remains unverified. With the advancement of drone technology, drones with wide-angle cameras with multiple axes of rotation are already available to overcome the limitations of a fixed camera drone [15].

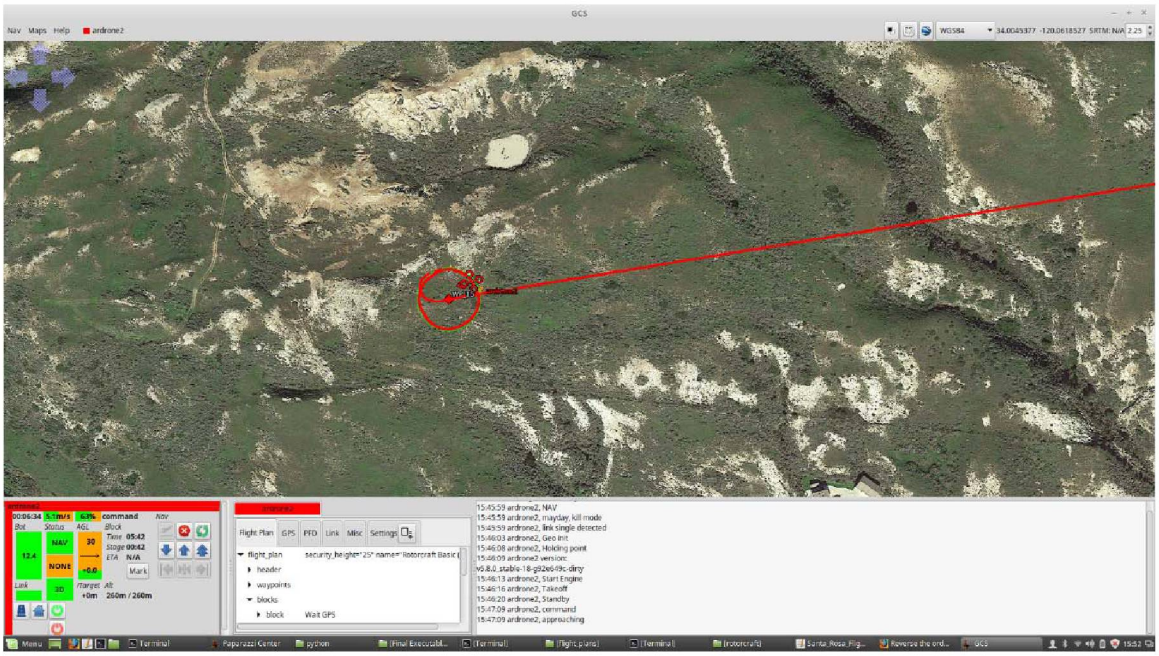

Figure 19. Drone circling the target location.

5.7 Event Handling Verification Results

Validation of the event system required launching the backend server developed from previous research work [16], which was the challenging part of verifying this aspect of the research. The provided source files require Pycharm to run, as it integrates the Django framework with the IDE. Pycharm can launch the website front end and server backend, and provides an interface to communicating the server.

Upon successfully launching the server, the source code was modified to add a new event handler for launching the drone, which calls the shortest path finding Python script, and trigger drone flight over the network. The script was launched with the longitude and latitude coordinates from the node specified by the event. Paparazzi was setup to listen for the shortest path to be received, and launching the drone was the indication that this verification was successful.

Additional work could be done to make the modifications more robust and maintainable, as the placement of the script for this test was put in a global path for the purposes of verification, but integrating it more closely with the project code should be considered.

Chapter 6: Conclusions

This thesis was to prove that the feasibility integrating autonomous flights in response to system events with the CI Rainbow project is possible, would provide a useful platform for the collection of data on Santa Rosa Island.

Functional requirements for this research to be considered a success were discussed in the introduction, and explored through the subsequent chapters to come up with experiments to confirm that the work done on this project accomplishes the set goals required for deployment in the larger CI Rainbow wireless sensor network.

The requirement that the drone is to be capable navigation was achieved by attaching the GPS device to the drone, connecting it through the USB interface with a serial cable, and verified by plotting its location as it was driven in a car.

Communication requirements were met by the onboard Wi-Fi on the AR Drone 2.0, and utilized by Paparazzi to command the drone at a high level to fly autonomously. This was verified to have been achieved by numerous test, both with indoor physical flights, and simulated flights commanded over a wireless network.

Remote sensing was achieved by the onboard camera, which stores data directly onto the device's memory. This was verified to be working out of the box, as it was a feature of the drone.

Autonomy was achieved by calculating the shortest flight path before flight began, and utilizing the Paparazzi UAV software to provide autopilot functionality on the drone. Simulated flights were used to verify this work, as physical flights were not possible due to restrictions in place at CI at the time of this writing.

The data management, user interface, static nodes, and event framework were implemented by prior research [16], and the integration with the drone was achieved by a modification of the source code, and validated by triggering a simulated flight through the event system.

With all requirements being achieved, it is asserted that this thesis has been appropriately supported by the work described and completed, and can be used on Santa Rosa Island to the benefit of students and researchers on and off the island. Naturally, it would require proper field testing in a controlled setting beforehand.

It is regrettable that there are legal difficulties with getting a permit to fly a drone over a national park. The logistics for doing so are difficult to resolve, such that the actual system could not be tested together to evaluate how it would perform if deployed on Santa Rosa Island with a small scale wireless sensor network supporting it. There are likely additional challenges to overcome as more systems are integrated together which would become apparent through testing, however, I am confident that no difficulty would prove to be insurmountable. As drones become more ubiquitous and their value becomes more recognized by society, the legal difficulties with flying will be resolved, and this body of work can be further developed, and deployed on Santa Rosa Island.

Chapter 7: Future Work

7.1 Evaluation

Any future researcher should seek to reproduce the results of this research by individually verifying the requirements of the system are met in the same way that this thesis has documented. Once this is accomplished, integration tests should be done to confirm that the drone will safely fly in the environment of Santa Rosa Island in response to system events, and that being disconnected from the wireless sensor network does not cause any issues.

7.2 Dynamic Planning

The work done for this thesis aimed to maximize features by relying on the core of functionality provided by the available Paparazzi software. One effect of this was that instead of performing the mapping and shortest path calculation onboard the drone, paths were statically calculated before the flight begins on the laptop. Only the waypoints of travel were sent, and the drone followed the path without further verification of its safety. Future work could be done to move this calculation to the drone such that it is able to handle starting from an unknown location, and finding its way to a destination. With better drone technology, some level of mapping could be done using cameras and other sensors to identify the drone's surroundings.

7.3 Upgrade Drone Hardware

While conducting this research, new and exciting UAVs were being released in the consumer market at a similar price point that the AR Drone 2.0 had, but with far more advanced features, such as multiple axis camera control, built-in GPS, more accurate sensors, and more robust flying power and battery duration. The work with the AR Drone 2.0 was done in the Paparazzi UAV framework such that it would be easy to replace the drone model used for this project with minimal effort required. Paparazzi is an open-source project which is constantly under development, adding new features and drone models to its supported hardware base, with an easy interface for developers adding support to any future drone models that can have its firmware overwritten by the ground control station.

7.4 Deployment

Ultimately, the goal of this research is to pave the road for putting autonomous drones on Santa Rosa Island, and once CSUCI can proceed with the CI Rainbow Project, several issues will need to be resolved for the project to work. With the AR Drone 2.0, the drone will need to be physically plugged into a power source to charge, whereas, in the future it is imagined that there will be charging platforms for the drone to dock and charge itself automatically.

Another area of future work to be investigated when deploying is the addition of collision avoidance mechanisms to protect the campus's investment in the hardware. The technology for implementing this is not yet available at a reasonable price point, but with the quality of market offerings improving at such a rapid pace, it will be likely this can be explored as it becomes more affordable.

7.5 Advanced Research

When autonomous drones can be deployed, and the overall system is confirmed to be working and providing data to researchers, this will open the doors to more advanced research in the field of autonomous drones. Neural network controlled autonomous flights, target acquisition and tracking, battery life estimation and management, and autopilot tuning in a real environment become very interesting areas of research that are available at this point, even if they are unnecessary to carry out the mission at hand. Consider the possibilities if drones can predict events before they occur, and roam the island indefinitely. It would provide perspective of conducting research on the island enabling innovative studies that would not be possible otherwise.

Chapter 8: References

[1] "Commercial Drones - A Global Strategic Business Report," Global Industry Analysts, Inc., January 4 2016. [Online]. Available: http://www.strategyr.com/pressMCP-7951.asp. [Accessed February 5 2017].

[2] B. Coxworth, "Vertex hybrid drone combines hovering and fixed-wing flight," NEW ATLAS, 23 April 2015. [Online]. Available: http://newatlas.com/vertex-hybrid-drone/37159/. [Accessed 5 February 2017].

[3] B. P. TICE, "UNMANNED AERIAL VEHICLES: THE FORCE MULTIPLIER OF THE 1990s," Spring 1991. [Online]. Available: http://www.airpower.maxwell.af.mil/airchronicles/apj/apj91/spr91/4spr91.htm.

[4] AMAZON.COM, "Prime Air," Amazon.com, [Online]. Available: https://www.amazon.com/Amazon-Prime-Air/b?ie=UTF8&node=8037720011. [Accessed 5 Februrary 2017].

[5] CIRES, "UAV flights," 5 February 2017. [Online]. Available: http://ciresblogs.colorado.edu/antarcticuavs/2017/01/22/uav-flights/.

[6] A. Momont, "Ambulance Drone," TUDelft University of Technology, [Online]. Available: http://www.io.tudelft.nl/onderzoek/delft-design-labs/applied-labs/ambulance-drone/. [Accessed 5 February 2017].

[7] V. Kumar, "Key Components of Autonomous Flight," Coursera, [Online]. Available: https://www.coursera.org/learn/robotics-flight/lecture/FuRZu/key-components-of-autonomous-flight. [Accessed 5 February 2017].

[8] C. Knospe, "PID Control," *IEEE Control Systems Magazine,* vol. 26, no. 1, pp. 30-31, 2006.

[9] Dictionary.com, "autonomy," Collins English Dictonary - Complete & Unabridged 10th Edition., [Online]. Available: http://www.dictionary.com/browse/autonomy. [Accessed 5 February 2017].

[10] S. Shair, J. H. Chandler, V. J. GonzÁlez-Villela, R. M. Parkin and M. R. Jackson, "The Use of Aerial Images and GPS for Mobile Robot Waypoint Navigation," *IEEE/ASME Transactions on Mechatronics,* vol. 13, no. 6, pp. 692-699, 2008.

[11] Adafruit, "Anemometer Wind Speed Sensor w/ Analog Voltage Output," [Online]. Available: https://www.adafruit.com/products/1733.

[12] E. Dijkstra, "A Note on Two Problems in Connexion with Grpahs," *Numerische Mathematik 1,* vol. 1, no. 1, pp. 269-271, 1959.

[13] L. Dà-Jiāng Innovations Science and Technology Co., "Phantom 3 Professional - Let your creativity fly with a 4K camera in the sky.," Dà-Jiāng Innovations Science and Technology Co., Ltd, 2017. [Online]. Available: https://www.dji.com/phantom-3-pro. [Accessed 24 March 2017].

[14] K. Scrivnor, "CI Rainbow: A Flexible WSN for Environmental Data," December 2016. [Online]. Available: http://hdl.handle.net/10211.3/184081 . [Accessed 24 March 2017].

[15] FTDI, "TTL to USB Serial Converter Range of Cables Datasheet," 23 May 2016. [Online]. Available: http://www.ftdichip.com/Support/Documents/DataSheets/Cables/DS_TTL-232R_CABLES.pdf. [Accessed 5 February 2016].

[16] M. TUDelft, "ARDrone Getting Started," TUDelft University of Technology, 16 October 2014. [Online]. Available: https://www.youtube.com/watch?v=eojAPZvT1Ck. [Accessed 16 October 2014].

[17] Boeing, "V-22 Osprey," Boeing, [Online]. Available: http://www.boeing.com/defense/v-22-osprey/. [Accessed 5 February 2017].