ASHESI UNIVERSITY COLLEGE

THE DRONE TOUR GUIDE

APPLIED PROJECT

B.Sc. Computer Science

Kojo Nyamekye Anyinam-Boateng

2018

ASHESI UNIVERSITY COLLEGE

THE DRONE TOUR GUIDE

APPLIED PROJECT

Applied Project submitted to the Department of Computer Science, Ashesi University College in partial fulfillment of the requirements for the award of Bachelor of Science degree in Computer Science

Kojo Nyamekye Anyinam-Boateng

April 2018

Declaration

I hereby declare that this Applied Project is the result of my own original work and that no part of it has been presented for another degree in this university or elsewhere.

Candidate's Signature:

…….

Candidate's Name:

…….

Date:

…….

I hereby declare that preparation and presentation of this Applied Project were supervised in accordance with the guidelines on supervision of Applied Project laid down by Ashesi University College.

Supervisor's Signature:

…….

Supervisor's Name:

…….

Date:

…….

Acknowledgements

Without God, Dr. Ayorkor Korsah, my family and friends this applied project would not have been a success. In view of this, I would like to say a big Thank you for all the support, guidance and care.

Abstract

This paper examines the implementation of a navigation and tour planner module using a drone as a tour guide. The project is a buildup on a subsequent project during a robotics class project at Ashesi University College. This was to aid the admissions team reduce pressure on Student Ambassadors and to also have tour guides available at all times.

This paper describes the implementation of the navigation and tour planning modules. Planning a tour is formulated by the Wavefront Algorithm. Also described in this paper, is how Dijkstra's shortest path algorithm is applied in conjunction with permutations to develop a tour planning module. Proper testing and evaluation were done to check the validity of the system.

Table of Contents

List of Tables

List of Figures

Chapter 1: Introduction and Background

1.1 Introduction

Over the years, the volume of visitors that book tours to the Ashesi University College campus have increased. Currently, all campus tours are led by a group of students (Student Ambassadors) who work for the admissions office. An interview with Miss Zeina Kowalski, the Senior Admission Officer at Ashesi University College revealed some challenges faced due to the present setup. Below are some challenges visitors and the student ambassadors face:

- Tours can only be given when a Student Ambassador is available. This is problematic because visitors might come at the time Student Ambassadors have classes or are caught up in their work.
- Visitors sometimes come in later or earlier than the time communicated to the admissions team.
- There is also the challenge of communicating effectively with foreign visitors who do not understand English during the tour.

1.2 Aim

The main aim of this capstone was to build a dynamic navigation module for a drone tour guide which could be used across the entire campus of Ashesi University College. This module is user-friendly and would require little effort and input from its users. The navigation module will help in the next development phase of the drone tour guide.

1.3 Related Work

Chief executive officers (CEO) and administrators among others seek to reduce cost while increasing efficiency at their workplace. As such, organizations are always looking

for ways to automate their process to introduce consistency and accuracy while reducing cost. This applied project is therefore not the first of its kind. Robot tour guide systems have been employed in different areas such as museums, schools, factories, tourism and more (Ivanov, Webster, & Berezina, 2017). Each of these sectors tailor their tour guide to suit their style of work and environment.

Within a six-day testing period at Deutsches Museum Bonn, a robot tour guide, RHINO was deployed to guide hundreds of visitors (Burgard, et al., 1998). This was made possible through a low-level probabilistic reasoning with high-level problem solving embedded in first-order logic. In order for RHINO to navigate its environment, it first localized itself within a map and then mapped out its environment to avoid obstacle collision. The tour guide had two user interfaces: an onboard interface and web interface. The onboard interface aided the visitors to interact with the tour guide system. This helped the robot know what the visitors wanted to see in the museum and effectively plot a path. The web interface provides a live feed of tours and documentation of the project (Burgard, et al., 1998).

Another museum tour guide, which aided fifty thousand people within a period of two weeks was called Minerva (Thrun, et al., 1999). Minerva is successfully deployed in the Smithsonian's National Museum of American History and made up of approximately 20 distributed modules which communicated asynchronously. At the basic level, the modules interact with the sensors on the robot as well as the effectors. Similar to RHINO, Minerva first localized, then mapped to avoid a collision and finally planned a path to be able to navigate its environment. This automated tour guide interfaced with its visitors through an improved web interface as compared to RHINO (Thrun, et al., 1999).

A common factor in the design of these robot tour guides is their ability to map out their environment and build an effective navigation module for the algorithm designers and

programmers. This allows them to easily move from one point to another while taking people on tours.

1.4 Background

The concept of a robotic tour guide for Ashesi was first explored in a robotics class project in 2013. This project started off with a robotic tour guide developed using the Turtlebot II robotics platform; a robot that navigated campus to perform tours which was worked on during the summer of 2014 by Delali Vorgbe and Wumpini Hussein. Further on in the Spring semester, Sheamus Yebisi'16 worked on the navigation module of the robot tour guide as his capstone project.

1.4.1 Tools

The tools and equipment used for the implementation of this original tour guide system comprise two main devices namely the Turtlebot II and the Android-based tablet.

The Turtlebot II is a programmable robot that is made up of a Kobuki base, an Xbox Kinect sensor and a workstation. The Turtlebot II's workstation runs on Robot Operating System (ROS). With the help of the Xbox Kinect sensor, the Turtlebot II can map out its environment to aid in effective navigation.

Figure 1.1 Turtlebot II

In order for users to interface with the Turtlebot II, an Android application was built to run on the Android tablet to help the tour guide users interact with the tour guide system. The application on the Android tablet communicates with the Turtlebot II via sockets.

Figure 1.2 Android tablet

1.4.2 Implementation Status and Shortcomings

With the current phase of the robot tour guide in Ashesi University College, the tour guide is able to move from a determined start location to an end location given that the angle of inclination is not too steep to hinder the movement of the robot tour guide. Also, a basic user interface was built to help users choose their start and end locations in order to commence the tours. Within the tour guide system, the Android application module has the ability to play pre-recorded audio information about locations on the Ashesi University College campus.

1.5 Problem Statement

On the Ashesi University campus, there is usually a disparity between the time schedules of the student ambassadors and the visitors. This disrupts the tour experience of the visitors on our campus. If enough attention is not given to this problem, the University risks losing potential students and donors, not forgetting the image of the school which is

also at stake. The robotic tour guide method will be explored to help improve the efficiency of the Admissions Tour team. The goal is to have a smooth tour guiding the process from the arrival of visitors, effectively communicating with them throughout the tour till their exit.

1.6 Proposed System

Given that problems faced by the Ashesi Admission team were steadily rising, this project seeks to provide an automated system that helps with the campus tours in Ashesi University College which are currently being conducted by Student Ambassadors.

The robot tour guide that was being developed was not able to move from one point to another if the inclination was very steep. From these observations and conclusions gathered from the Turtlebot II, this project sought to eliminate that problem by introducing the use of a drone as the tour guide. Since the drone was sky bound and the problem at hand were tours, the name "Sky-Tour" for the project came about.

The Sky-Tour system in Ashesi University College would help the admission team to become more effective and consistent in the execution of their task without eliminating the old system of using tour guides. Sky-Tour hopes to incorporate the current system in place to be able to still have the human touch. This addition will be achieved by allowing the tourists to choose between Student Ambassadors or the Sky-Tour. However, the Sky-Tour will be used solely when there are emergencies. A practical example of this is if a tour is scheduled for 9:00 am to 10:00 am, yet the visitors arrive late at 9:30 am displacing the Student Ambassadors and the Admissions tour team who may have other tasks at hand. In such a circumstance, the Sky-Tour could be brought in to control the situation.

In addition, after a careful review of the work that was done in the past, a major setback of the Turtlebot II was observed. This setback was the inability to climb or descend

the staircases on campus. As such, the Sky-Tour system will eliminate this problem with ease since drones will be used.

1.7 Overview of Chapters

This report is broken into six chapters. In Chapter 2, the readers are made to understand the system requirements of the project and how requirements are gathered and classified. In the third chapter, it describes a high-level overview of the system architecture and design. Chapter 4 describes the implementation of the system that was outlined in the previous chapter. After the implementation, Chapter 5 presents the tests that were conducted, and the results obtained. Finally, the report ends with conclusions and recommendations for further study in Chapter 6.

Chapter 2: System Requirement Specifications

In this chapter, the main aim is to provide a description of the functionality that will make up the Sky-Tour system. It provides the various scenarios in which the Sky Tour system will be of great help to the users of the system.

2.1 Project Overview

The Sky-Tour System consists of two main components namely: a web application and the Sky-Drone application. The web application servers as an interface between the user and the Sky-Tour System. All user interactions with this system are done with the Sky-Tour Web Application. The Sky-Drone application would serve as the backbone for the entire system. It is made up of all the path planning, routing and navigation algorithms responsible for a tour to take place. In a nutshell, these two components depend on each other to make up a complete tour guide system.

2.2 User Classes and Characteristics

The Sky-Tour system is a system designed to aid the campus Admissions Tour team in Ashesi University College with conducting tours for visitors and potential students. After interviewing some stakeholders and from observations made of this process, it was noticed that the system would be designed for two categories of people namely the visitors and the administrators. The visitors are people who come to Ashesi University College with the aim of gathering information about the university and the administrators are the group of individuals that manage the systems and make sure tours run smoothly.

2.2.1 Use-Case Scenarios

Scenario 1 – The Visitors without SkyTour

Zoe and her friends are fond of Ashesi University College and have always wanted to come and pursue an undergraduate study there. In light of this, they decided to take a trip to Ashesi University College. After they signed in at the security gate and found a parking space, they were directed to the reception where they met Eric, the receptionist. After they asked Eric a couple of questions, they requested for a campus tour. Unfortunately, they came at a time no Student Ambassador was available to take them on this tour. However, they had two options: either they stay and wait for an available Student Ambassador or go back home. They ended up waiting and finally got to explore Ashesi's campus and had their questions answered. This tour was however shortened to 15 minutes as so much time was spent waiting.

Scenario 1 – The Visitors with SkyTour

In order to avoid keeping Zoe and her friends waiting, the Sky-Tour was introduced. With the help of Sky-Tour, Ashesi would be able to reduce the waiting time for a visitor. As soon as a visitor walks in after a few questions have been asked, Eric would then set up the Sky-Tour System by providing the system some details. These details include: whether Zoe and her friends want a customized or full tour. In the situation in which they want a customized tour, they are made to choose if the customized tour should be generated based on time constraints or their own selection. Once this is done, they are given a tablet with a Sky-Tour Drone to guide them around campus. Once Zoe has a question, she logs it into the questions section and if the system has been asked a similar question she is provided an answer. On the other hand, if the system does not have a similar question logged, she would be given an answer once she gets back to the reception.

Scenario 2 – The Administrator

With the aid of Sky-Tour, the head of admission Dr. Araba Botchway is provided with some analytics that helps she and her team adjust the tours as time goes on. She realized that a lot of questions are asked anytime visitors get to the Founders Court area and because of that, she has asked for another pre-recorded video of the Founder's Court to be done with the aid of the questions people frequently ask.

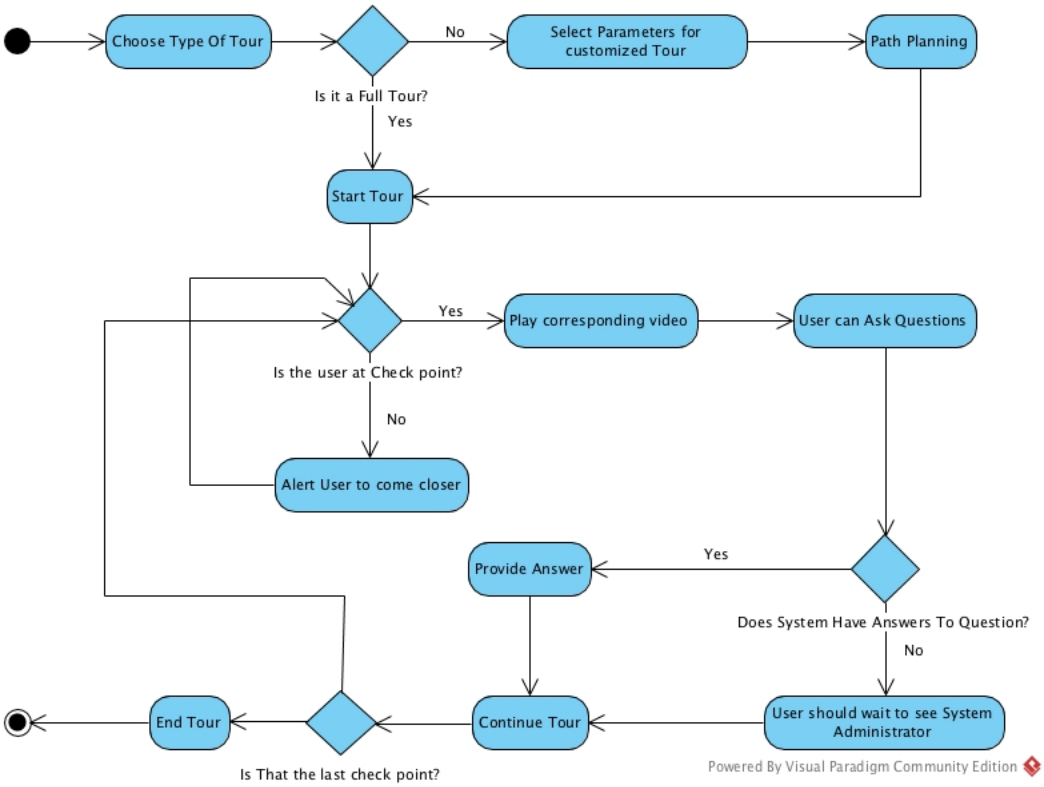

Figure 2.1 Tour system flow chart

2.3 System Features

The following are the list and brief descriptions of the various features and functionality of the Sky-Tour System. The system's features have two main divisions, namely the core features and the extra features. The core features are the heartbeat of the entire Sky-Tour system. This means that the system cannot do without them and must be

implemented to the highest capacity. The extra features are functions that are not of great importance to the system.

2.3.1 Core Features

1. Campus Tour

   The campus tours in the system are in two variations, that is a full campus tour and a customized campus tour.

   i) Full Campus tour

      When a user selects this option, he would be given a complete tour of the university. A full campus tour is made up of locations which are predefined by the system administrator.

   ii) Customized Campus Tour

      When a user selects this option, he would be given a list of locations on campus to selects where he wants to visit and based on an optimization algorithm, the shortest path is generated for the user. On the other hand, the user can provide a time frame he has, and a tour can be generated for him.

2. Analytics

   When the administrator logs into the system, he can see the various data gathered on each tour which would generate graphs among others to help the administrator make meaning out of it and aid him in making a constructive decision to help fine tune the system.

2.3.2 Extra Features

1. Student Ambassadors Scheduling

Some visitors may prefer human interaction and as such, the Sky tour system will be built and designed to aid in scheduling and managing human tour guides (Student Ambassadors) when visitors request for them.

2. Turtle Bot

   To enable the TurtleBot II to communicate to the Sky-Tour Application, a web socket is used. This will give room for more tour guides to be incorporated into it since the turtle bots are readily available.

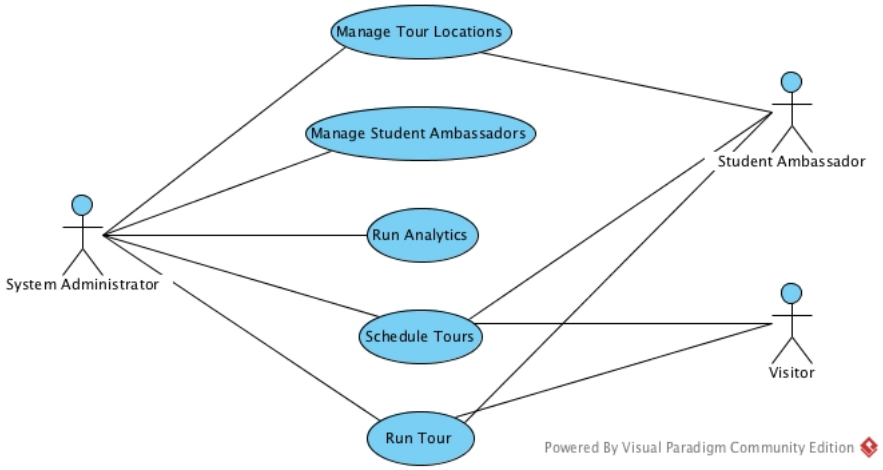

Figure 2.2 Use case diagram

2.4 Non-Functional Requirements

The major concern here with this system is security. Two aspects of security that are of particular concern to this system are structural security and tour guide security. Structural security basically deals with not allowing the drones and visitors to walk into restricted areas on campus or areas that are undergoing construction. Tour guide security also deals with putting a system in place that would prevent the loss of the tour guides and tablets since they are very portable and can be easily carried away by the visitor.

2.5 Tour Guide Drone

The main aim of the entire system is to aid visitors to tour the campus at ease without spending a majority of their time waiting. The Sky Tour System utilizes an Erle-Copter as its tour guide to show people around the campus. This involves movement from one coordinate to another. Also, the Erle-Copter should be able to transmit its current location to the system in order to help coordinate the tour.

Chapter 3: Architecture and Design

The architecture and design of a system or application is the blueprint of the system or application. The architecture and design show the team working on the system or application, how everything works and how the various modules are inter-related with each other. At this stage, the modules of the system are designed to meet its system requirements. On the other hand, the various modules that make up the system are structured in order to meet the non-functional requirements of the system. Sky-Tour has two components, the user interface and the drone.

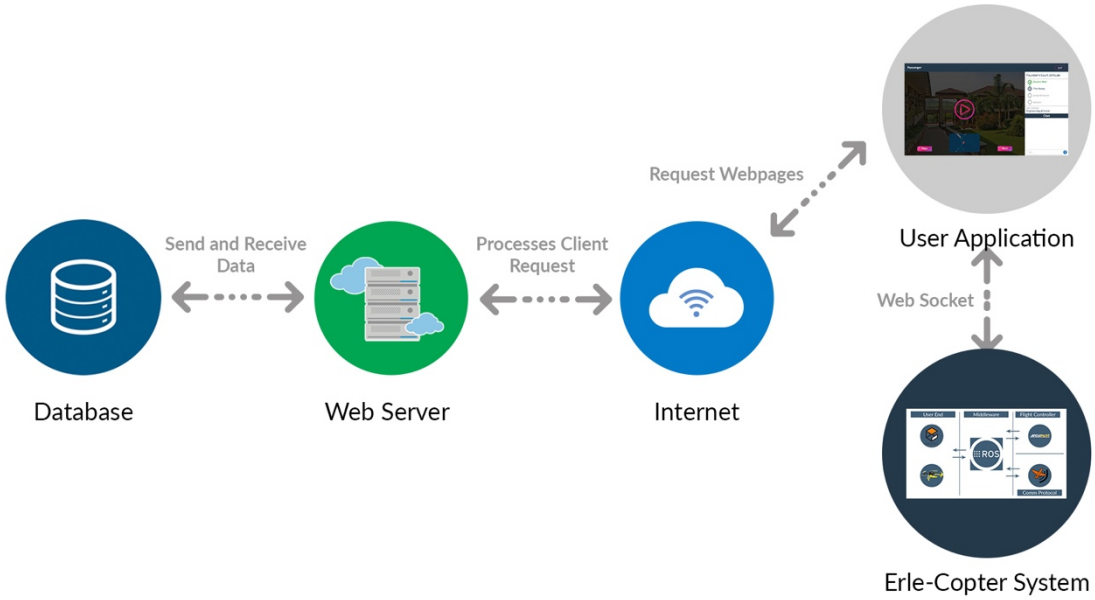

Figure 3.1 General system overview

To meet the system requirements, both functional and non-functional requirements as stated in Chapter 2, two main modules with their sub-modules have been properly structured in order to cater for these system requirements. The two main modules are the user interface and the tour guide.

3.1 User Interface Module

This is a web application that allows users (both administrators and standard users) to interact with the tour guide system. The architecture of the User Interface Module is based on the Model-View-Controller (MVC) approach that divides the application into three main logical components.

The model is the component responsible for all aspects of the application that has to do with data. The view is the component of the application that the user actually sees. That is the frontend of the application. The controller component serves as an interface between the model and the view. This is where all the business logic of the application is handled.

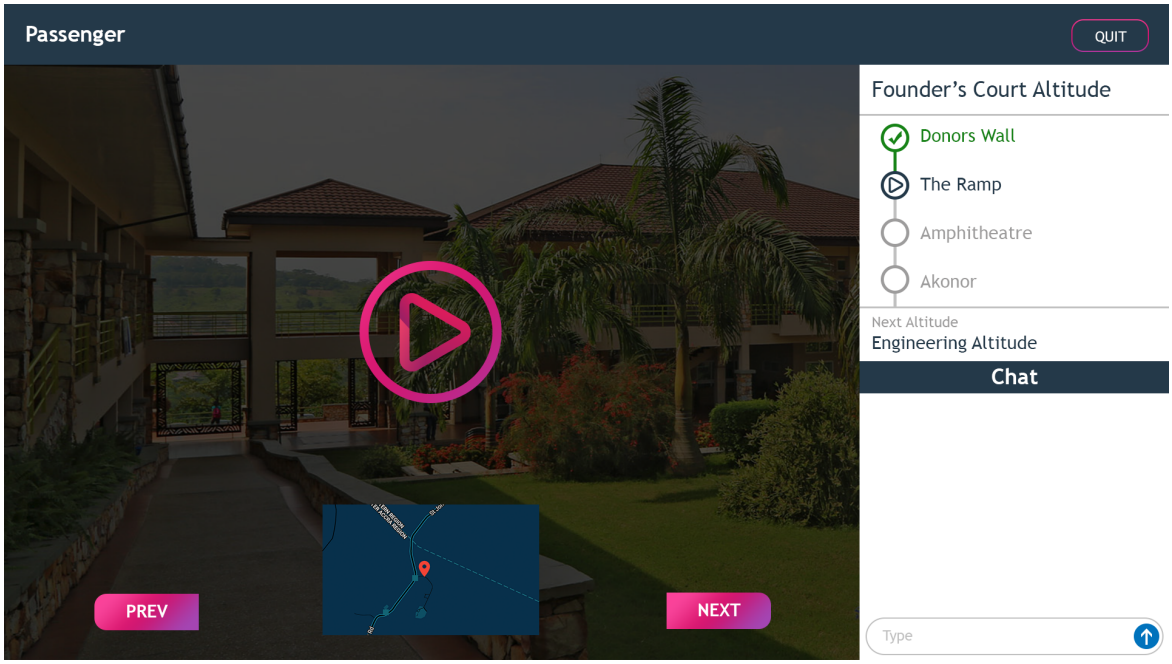

Figure 3.2 User interface (UI) design of user application

3.2 Tour Guide Module

The Tour Guide Module uses an already existing system architecture defined by Erle-robotics. The Tour Guide Module is made up of three main components which are the user end module, middleware module and controller module. The controller module is made up of two modules; the flight controller and communication protocol. The flight controller

and communication protocol provide a means of sending a set of instructions to either a physical robot or a simulated robot. The middleware, Robot Operating System (ROS) is a robotics framework that allows robotics developers write code with ease. With these modules integrated together, alongside a user interface module, the Sky-Tour is able to give tours to its users.

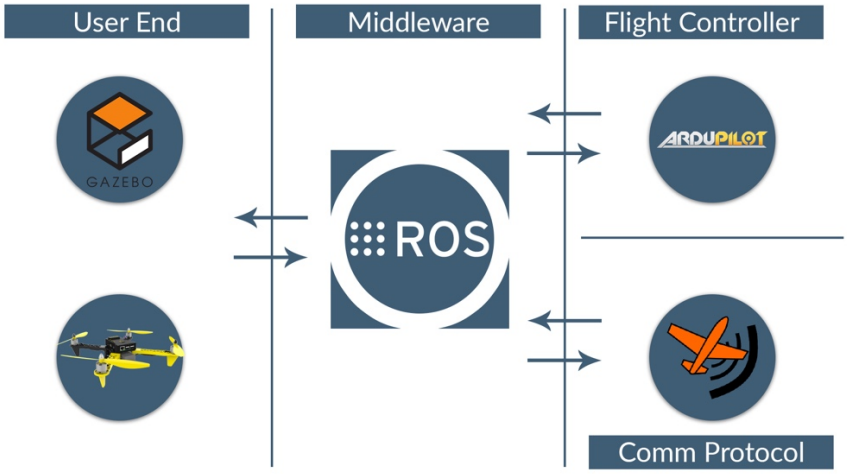

Figure 3.3 Tour guide system architecture

3.2.1 User End – Erle-Copter

The Erle-Copter developed by Erle Robotics is a smart quadcopter drone that runs on the Robot Operating System. This smart drone was specifically developed for developers to build applications. The Erle-Copter is made up of various components such as Erle-brain 3, DJI F450 frame, GPS, camera, LiPo Battery and Telemetry Link. Figure 3.4 shows the 3-D model of the Erle-Copter.

Figure 3.4 Erle-Copter [1]

3.2.2 Middleware – Robot Operating System (ROS)

Robot Operating System (ROS) is a collection of tools, libraries and conventions that aim to simplify the task of creating complex and robust robot behavior across a wide variety of robotic platform (Open Source Robotics Foundation, n.d.). The ROS system is designed such that each independent node represents a part of the robot or system. They communicate with each other by subscribing or registering to a topic or service and publishing messages to each other. All the publishing and subscribing are overseen by the ROS master that allows individual nodes to find and communicate with each other. Figure 3.4 shows an overview of how ROS nodes communicate with each other with the coordination of the ROS Master.

[1] Image obtained from http://docs.erlerobotics.com/erle_robots/erle_copter

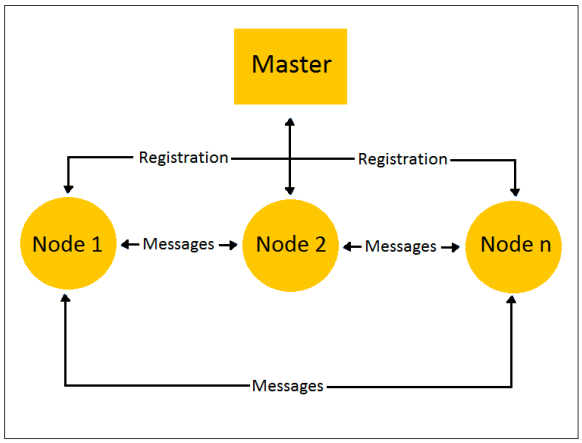

Figure 3.5 Overview of ROS publishing and subscription[2]

3.2.3 Movement Module

The movement module is a class responsible for the basic movement of the Erle-Copter. For example, moving the Erle-Copter from the Library to Lecture Hall 218 on the Ashesi University College campus. The movement module in conjunction with the navigation module will be able to move the Erle-Copter to give tours.

3.2.4 Navigation Module

The navigation module is responsible for planning a route from a given point of departure to a given point of arrival using the wavefront algorithm (Cober, Styler, & Naaktgeboren, 2006). The wavefront algorithm is a path planning and obstacle avoidance algorithm that allows the robot tour guide system to determine the optimal path of a tour giving the environmental constraints such as areas human beings cannot walk through and walls. This is done to help the drone not to just fly from one point to the other.

[2] Image obtained from http://docs.erlerobotics.com/robot_operating_system/ros/basic_concepts

3.2.5 Flight Controller

The flight controller the Erle-Copter uses is ArduPilot. ArduPilot is an autopilot software capable of controlling any unmanned vehicle system such as copters, quad copters, rovers, planes, submarines, antenna trackers among others (ArduPilot, 2016).

3.2.6 Communication Protocol

The Micro Air Vehicle link (MAVLink) is a communication protocol used by the unmanned vehicle to allow the unmanned vehicles to communicate with each other, an onboard software, ground station control, web application or mobile application. MAVLink is a header-only message library, meaning all the class and functions that make up the library are accessible to the compiler in the header files. With these characteristics, it makes MAVLink extremely lightweight (QGroundControl, n.d.).

3.3 Gazebo

Gazebo is an open source robotics simulation toolkit that allows roboticists to design, implement and test software, hardware and train artificial intelligent systems before deploying it for production use. This toolkit is essential in the field of robotics as it gives the ability to perform all sorts of tests on a simulated robot. This help roboticists reduce cost since a little amount of money will be spent in building a wrong robot or robot part in a simulation compared to actually manufacturing the wrong robotics part. In figure 3.5, there is a simulated Erle-Copter in gazebo (Open Source Robotics Foundation, 2014).

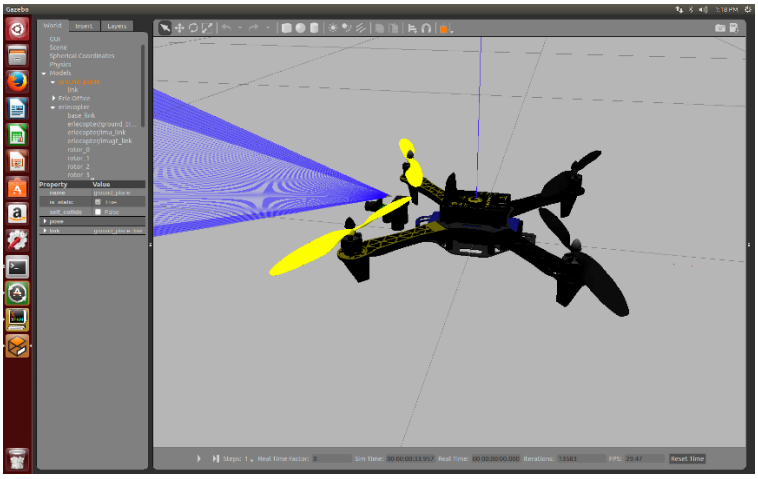

Figure 3.6 Simulated drone in Gazebo

Chapter 4: Implementation

This chapter examines some concepts, software, hardware and tools used to build Sky-Tour. For the scope of this project, the focus was on the navigation module of the system.

4.1 Erle-Copter

As described in Chapter 3, the tour guide is based on the Erle-Copter from Auctronic Robotics. This Drone was assembled from its constituent parts, namely electronic speed control (ECS) motors, DJI F450 frame, Erle-Brain 3, LiPo battery, Radio Control (RC) receiver, connectors and fasteners (bolts and nuts, screws and disposable restraints). Figure 4.1 shows the individual parts of the Erle-copter before assembly, while Figure 4.2 shows the assembled drone.

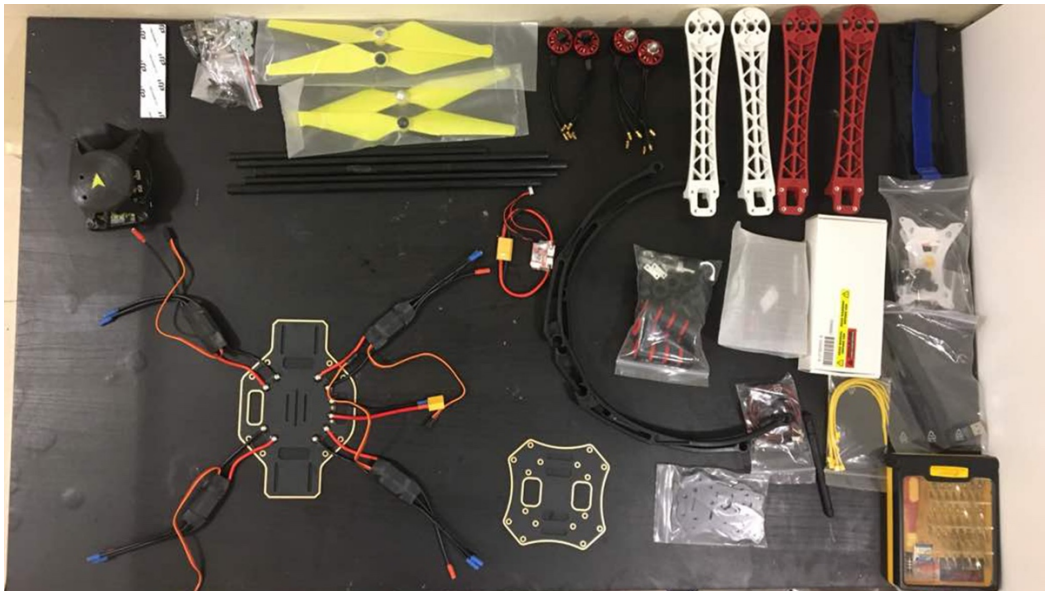

Figure 4.1 Parts of Erle-Copter

After the Erle-Copter was assembled, the various sensors (GPS, telemetry, barometer, accelerometer and compass) were calibrated with the help of QGroundControl (a flight control and mission planning software) (QGroundControl – Drone Control., 2017). This was then followed up with configuring the Wi-Fi and installing the ArduPilot. Through

a secure shell connection interfacing with the Erle-Brain 3 from a laptop, a ROS workspace on the drone was built.

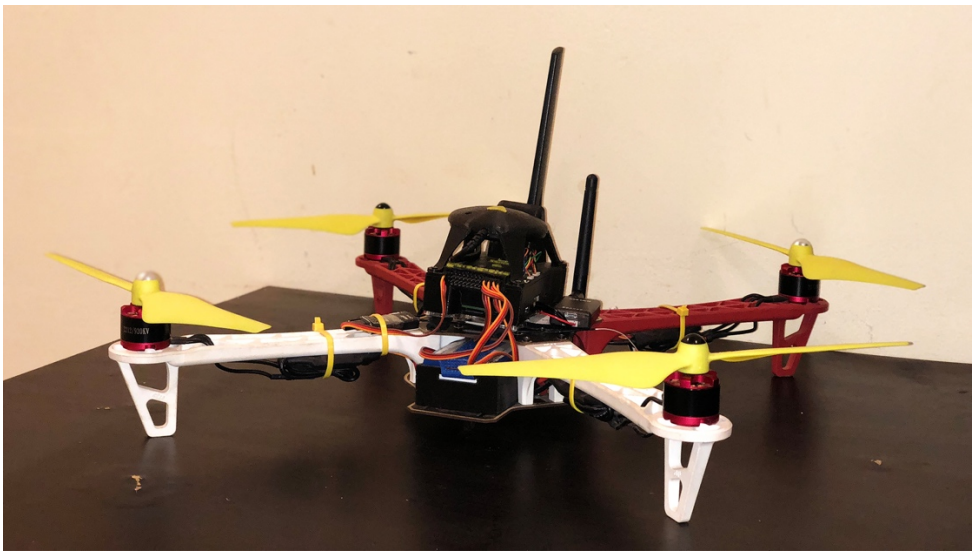

Figure 4.2 Assembled Erle-Copter

4.2 Gazebo

To avoid crashing the physical drone and saving money, a 3-D model of Ashesi University College in Gazebo was modelled to aid in simulating the drone tour guide. The 3-D model was developed form the architecture drawings (blueprint) which were obtained from the Operations and Facilities Management of Ashesi. All unwanted components of the image that were not needed were removed and the image converted to Scalable Vector Graphics (SVG) format, then to portable network graphics (PNG) format with Adobe Illustrator. The PNG file was then imported into Gazebo's Building Editor as the plan for the 3-D model and the building tools in Gazebo were used to build the 3-D model of Ashesi University College.

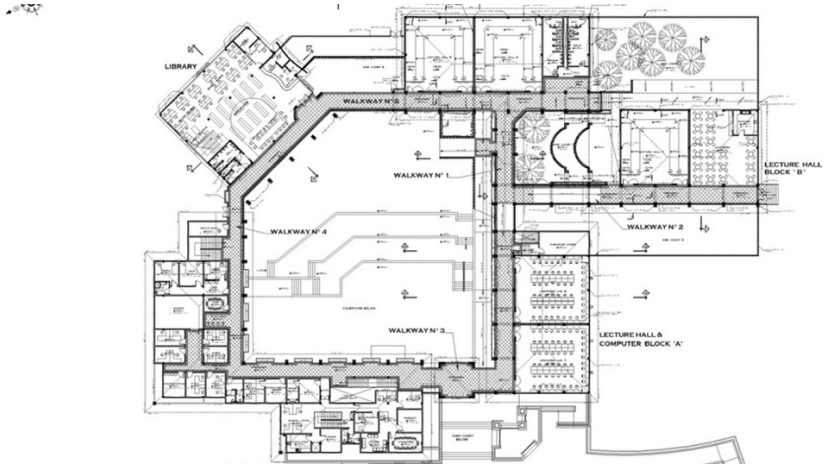

Figure 4.3 Blueprint of Ashesi University College

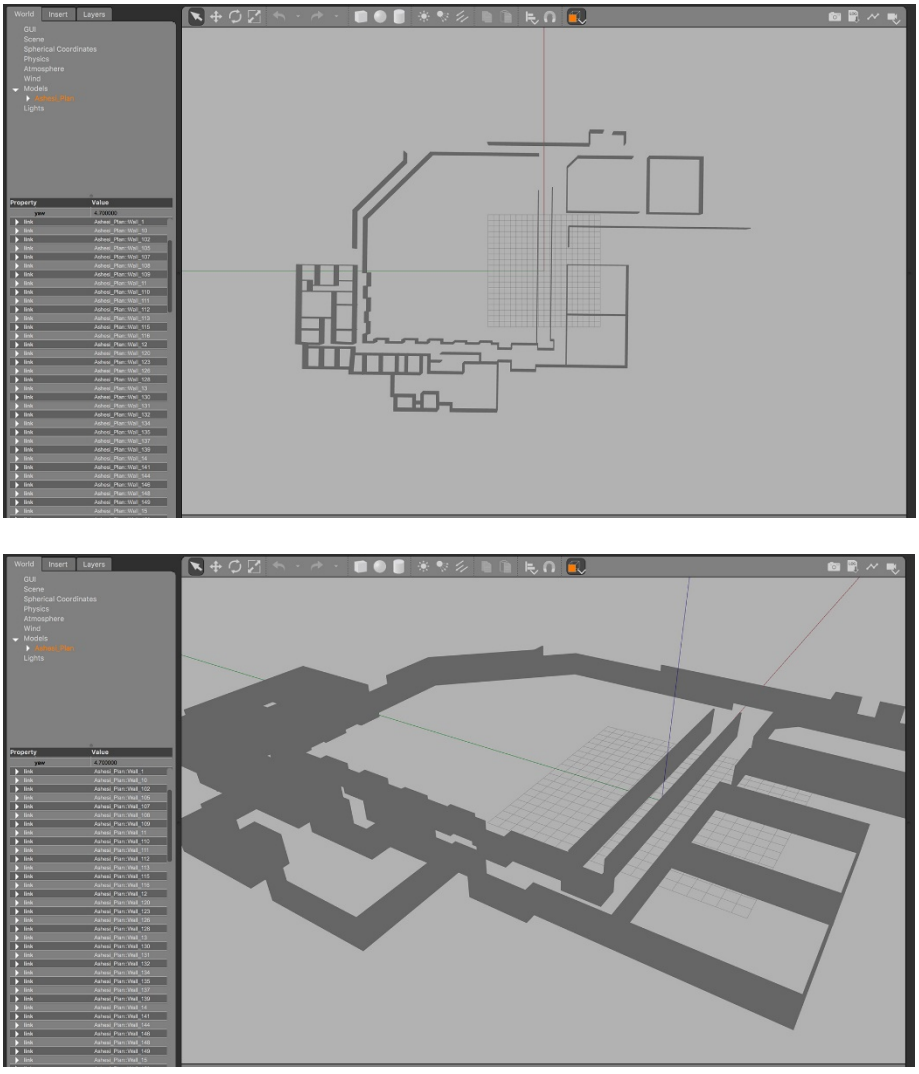

Figure 4.4 3-D model of Ashesi University College

From the Erle robot repository on GitHub, the 3-D model of the Erle-Copter was downloaded and imported it into Gazebo.

4.3 Tour Planner

In order for a tour to take place, the system takes in a start location and various locations a visitor wants to visit in a form of a list (array). The first index is the start location while the rest of the indexes are the various places of interest to the visitor. For the program to generate the most efficient route in terms of distance, it reads in a JSON file that contains all the necessary information needed for the planning.


```
{
    "Start" : {
        "left_node" : ["Library"],
        "right_node" : ["Lab 222"],
        "wf_length_to_left_node" : [99],
        "wf_length_to_right_node" : [10],
        "x_coordinate" : 66,
        "y_coordinate" : 54
    },
    "Lab 222" : {
        "left_node" : ["Start"],
        "right_node" : ["Lab 221"],
        "wf_length_to_left_node" : [10],
        "wf_length_to_right_node" : [16],
        "x_coordinate" : 66,
        "y_coordinate" : 63
    },
    "Lab 221" : {
        "left_node" : ["Lab 222"],
        "right_node" : ["Amphitheatre", "LH 217"],
        "wf_length_to_left_node" : [16],
        "wf_length_to_right_node" : [35, 36],
        "x_coordinate" : 51,
        "y_coordinate" : 63
    },
```


Figure 4.5 A snippet of JSON file of locations

Since the JSON file provides the left and right nodes of a parent node, a graph is built. Using the permutation formula $P = \frac{n!}{(n-r)!}$ the number of permutations is calculated and generated on the list while ignoring the first index.

Example

List = [“a” , “b” , “c” , “d”], since we ignore the first index (Starting Location), the list becomes [“b” , “c” , “d”].

n is the number of values in the list = 3.

Using $P = \frac{n!}{(n-r)!}$, where n = 3 and r = 3

$$P = \frac{3!}{(3-3)!}$$

$$P = \frac{3!}{0!}$$

$$P = 6$$

```
Number of permutations is 6. The permutations are [“b” , “c” , “d”], [“b” , “d” , “c”], [“c” , “b” , “d”], [“c” , “d” , “b”], [“d” , “b” , “c”] and [“d” , “c” , “b”]
```

Once the generation of the different permutations is done using the graph built previously, a Dijkstra’s algorithm was used to compute the shortest path between adjacent locations in each permutation, and hence the path for each permutation. At the end of this program, the optimal order in which the visitor should visit the various location has been generated.

4.4 Map Generator

For the map generator module, the most important element is the blueprint of the environment you wish to navigate. In this case is the blueprint of Ashesi University college (Figure 4.5.1 A). The program reads in the images and begins to draw a square grid of 10 pixels by 10 pixels on the images (Figure 4.5.1 B). Once this is done, the various 10 pixels by 10 pixels are then picked one after the other and the sum total of Red, Green and Blue (RGB) values are then computed. For a completely white area, the sum total of the RGB value is 765 and black is 0, while computing the RGB values at the various 10 pixels by 10 pixels, a map is being built with a list of lists where for every RGB value calculated which is less than 550 is marked by 1 (meaning there is an obstacle) in the list and on the other hand it is marked by 0. At the end of the algorithm, the image that was fed into the program should be represented by 1’s and 0’s (Figure 4.5.1 C).

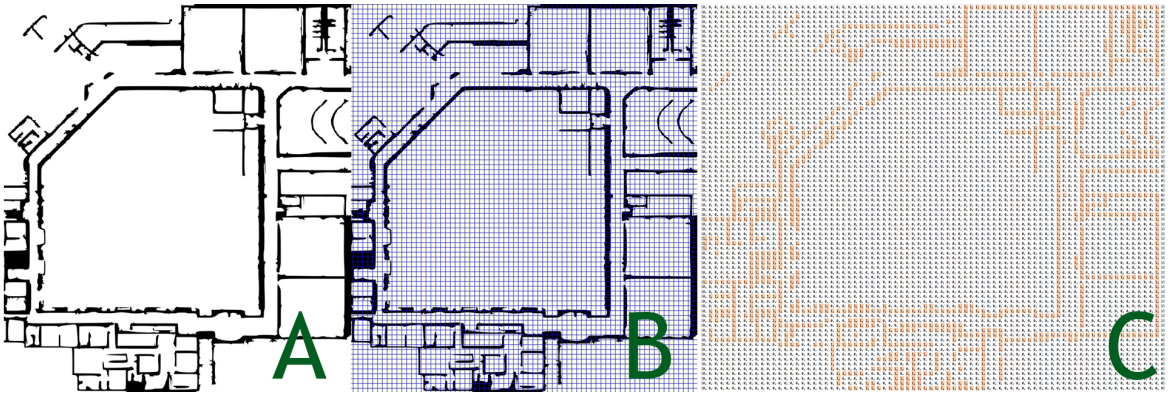

Figure 4.6 Image transformation to 1's and 0's

4.5 Route Planner

The wavefront algorithm was used in the route plaaner because of its ability to avoid obstacles (such as walls) and plan the shortest path from the start state to the goal state. The route planner uses the output generated from section 4.4 of this paper. The obstacles in the map are indicated by 1 as mentioned in section 4.4. In Figure 4.6 is a generated four-point wavefront algorithm. The obstacles, in this case, are the areas marked with 1's, the starting position is marked green while the goal is marked red. The path is formulated by counting down from the number at the goal state to the start state.

18	17	16	15	14	13	1	13	12	1
17	16	15	14	13	12	1	1	11	12
18	17	1	13	12	11	10	1	10	11
19	18	1	1	11	10	9	8	9	10
20	19	1	1	1	1	8	7	8	9
21	1	1	1	9	8	7	6	1	1
22	1	1	1	10	9	1	5	4	1
23	24	1	1	11	10	1	4	3	2

Goal Cell
Start Cell
Obstacle Cell
Shortest Path Cell

Figure 4.7 Generated Wavefront

In this example, the areas labeled with 1's are the areas the Erle-Copter cannot fly through. These areas marked with 1's include areas human beings cannot walk through and walls. The start position of the Erle-Copter is labeled 28 and the end position (goal) is

labeled 2. To achieve an optimal solution with respect to the algorithm being used, the Ashesi University College campus was divided into grids for this algorithm to effectively work.

How the Wavefront Algorithm Works

1. All obstacle cells are labeled with 1.
2. The goal cell is labeled 2.
3. The value in the current cell is taken, incremented by one and placed into the cells to the top, bottom, right and left of the current cell provided there is no value there.
4. Step 3 is repeated till the start cell is filled.
5. The shortest path is then generated from counting down from the start cell to the goal cell.

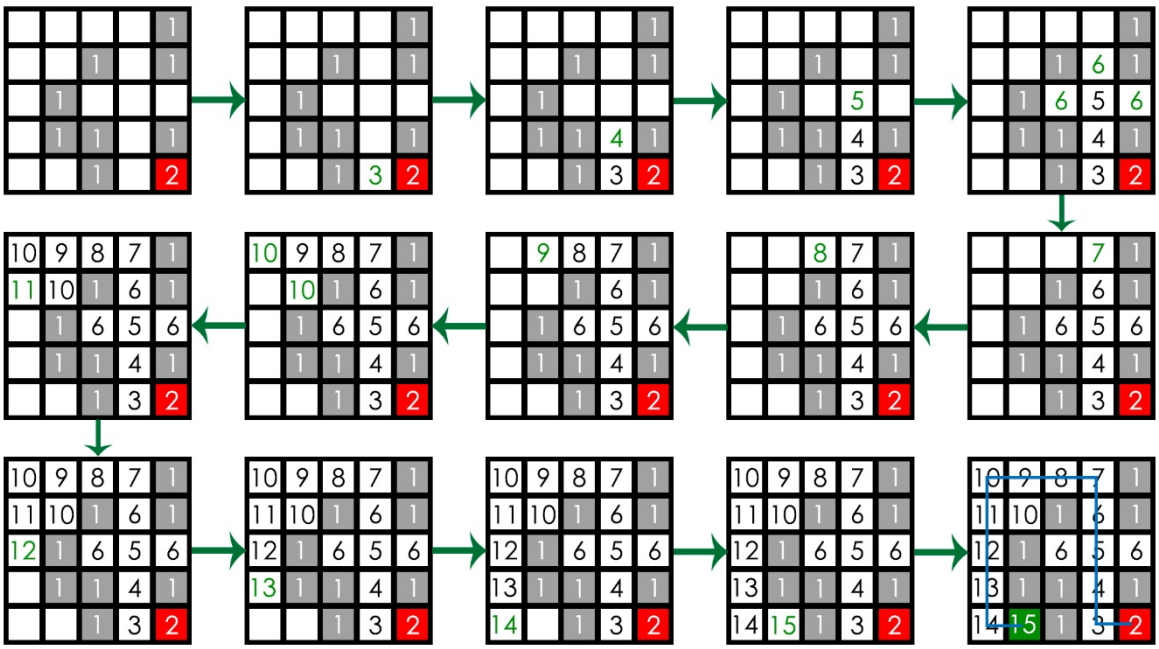

Figure 4.8 Step by step demonstration of Wavefront Algorithm

After the Tour Planner has generated the sequence of locations for the tours, the route planner takes in two parameters: the current location and the next location in order to generate the wavefront for the drone to navigate the campus. This action is repeated until the last but one location. During the generating of the shortest path, a movement pattern is

generated. The movement pattern is generated based on the current direction of the drone with respect to the physical world. The table below shows the behavior of the drone with respect to the physical world.

		Physical World Direction			
Drone Direction		**West [0]**	**North [1]**	**East [2]**	**South [3]**
	West [0]	Go Straight	Turn Right 90°	Turn 180°	Turn Left 90°
	North [1]	Turn Left 90°	Go Straight	Turn Right 90°	Turn 180°
	East [2]	Turn 180°	Turn Left 90°	Go Straight	Turn Right 90°
	South [3]	Turn Right 90°	Turn 180°	Turn Left 90°	Go Straight

Table 4.1 Drone direction in relation to physical world direction

4.6 Movement

In the implementation of the movement module, a generic drone movement class was created. The generic class was used to enable the system to use multiple drones from different vendors. Various functionality were implemented to enable the drone to:

- take off and land.
- turn left and right.
- move forward and backward.
- move to a specific altitude.
- Arm and disarm.

Chapter 5: Testing and Results

To be able to validate the functionality of the navigation module developed for the drone tour guide, tests were conducted. This chapter describes the tests that were conducted, proof of test and result derived.

5.1 Testing the Blueprint Grid functionality

This test was performed to check the validity of the blueprint grid functionality. This module of the system takes in either a portable network graphic(PNG) image or a joint photographic expert group (JPG) image. The algorithm then grids the image into 10 pixels by 10 pixels. The left image of Figure 5.1 shows the input joint photographic expert group (JPG) image and to the right of Figure 5.1 shows the output.

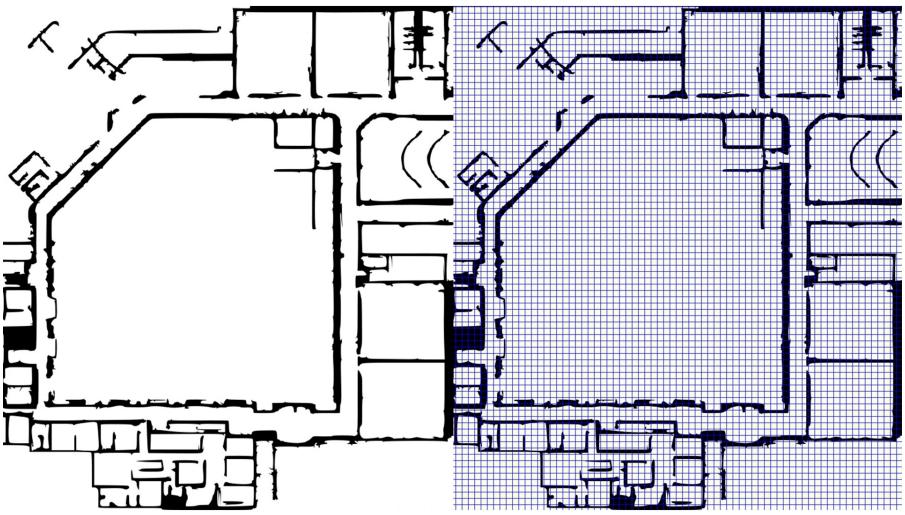

Figure 5.1 Test of blueprint grid functionality

Based on the input image and the output image obtained from this functionality, it can be concluded that the blueprint grid functionality is valid.

5.2 Testing the Map Generator functionality

This test was performed to check the validity of the map generator functionality. This module of the system is to take in the image generated from the blueprint grid

functionality as its input. This algorithm processed the image by taking a cell and calculating the red, green, and blue (RGB) and comparing it to a threshold of 550. If the cell was below the threshold, it was marked as 1 or it was marked as 0 if it was above 550. The left image in figure 5.2 shows the list of arrays generated to form the map. In the left image of 5.2, all 0's was removed to give a clear picture of the map obtained.

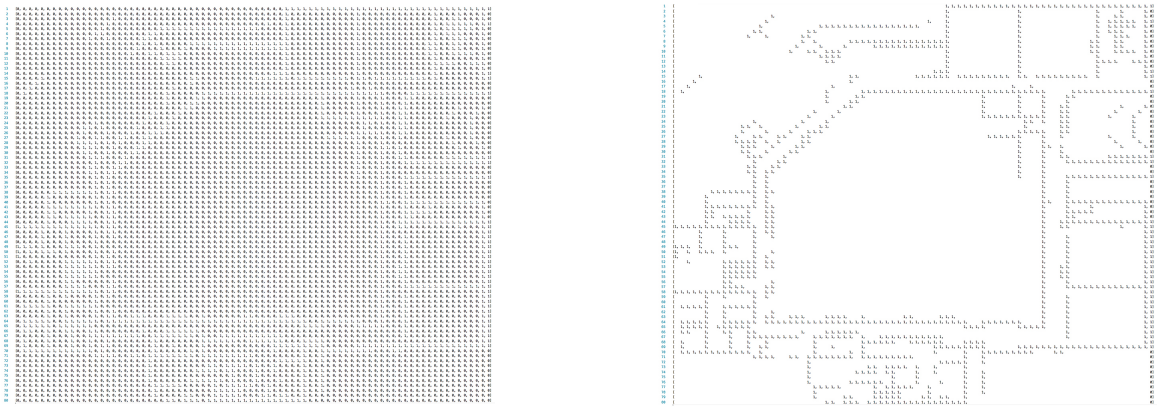

Figure 5.2 Test of map generator functionality

Based on the input image, the output list of arrays obtained from this functionality and also comparing the map generated to the original blueprint image that served as in input in the blueprint grid functionality, it can be concluded that the blueprint grid functionality is valid.

5.3 Testing the Route Planner functionality

This test was performed to check the validity of the route planner functionality. This module of the system is to take in the map generated from the map generator functionality, a departure and arrival location and performs a wavefront algorithm to find the optimal path. This module reads in a JSON containing all the details of the locations of the map. With this, the user just specifies where he wants to depart and arrive. The JSON file allows the application to know the X and Y coordinates of the locations specified.

Figure 5.3 shows the wavefront path generated for a user who wants to move from the Amphitheatre to Lab 221. The block of JSON object in the red rectangle in Figure 5.3 shows the location details of the Amphitheatre and Lab 221

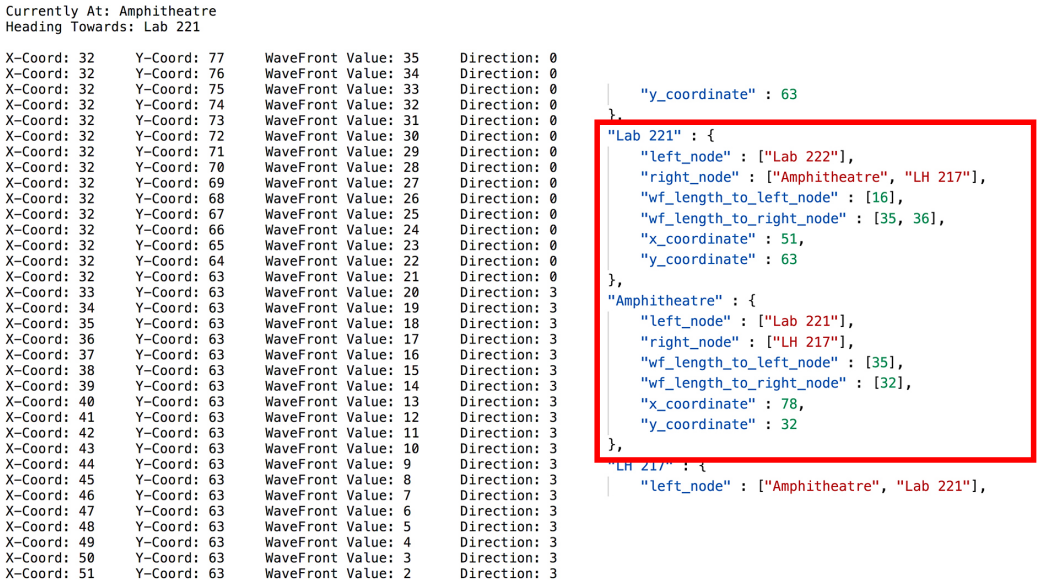

Figure 5.3 Test of route planner functionality

Looking at the location details from the JSON object of both the Amphitheatre and Lab 221, it was observed that the Y-coordinate of the Amphitheatre in the route generated did not correspond to the location details of Amphitheatre the JSON object. This was because the wavefront algorithm executed one iteration before printing out the values hence offsetting the Amphitheatre's Y-coordinate by one. At the end of the wavefront algorithm, it was observed that the X and Y coordinates of the arrival location that was generated corresponded to the location details of Lab 221 in the JSON object

From these results and observations, it can be concluded that the route planner functionality of the system is valid.

5.4 Testing of Movement Pattern Generator Functionality

This test was performed to check the validity of the movement pattern generator functionality. This module takes in the route generated from the route planner functionality

as its input. Based on its previous and current location, it generates a movement pattern for the drone. Figure 5.4 shows a movement pattern generated for a tour from the Amphitheatre to Lab 221.

```
[
    {
        "0": 15
    },
    0,
    {
        "3": 19
    }
]
```

Figure 5.4 Test of movement pattern generator functionality

Looking at the output generated, it was observed that the file contained two main categories of values: a single value and a dictionary. In Figure 5.4, the single value is 0 and the dictionaries are {“0”: 15} and {“3”: 19}. The value of the dictionary tells the drone how long it should move forward while the single values tell the drone in what direction it should turn. Table 5.1 shows interpretations of the single values.

Single Value Interpretation	
Single Value	**Interpretation**
0	Turn Left
1	Move Forward
2	Turn Right
3	Turn 180°

Table 5.1 Single value interpretation

From our test results, we can say, the drone is to:

- Move forward for 15 cycles.
- Turn left.
- Move forward 19 cycles.

Based on the input from the route planner functionality and the output derived obtained from this functionality, it can be concluded that the movement pattern generator functionality is valid.

5.5 Testing the Tour planner functionality

This test was performed to check the validity of the tour planner functionality. This module takes in an array of locations with index 0 as the start location. Ignoring the value of index 0, the permutations are generated on the rest of the values. The Tour is planned by computing the cost of the various permutations using Dijkstra's shortest path algorithm. The least cost is selected as the tour plan.

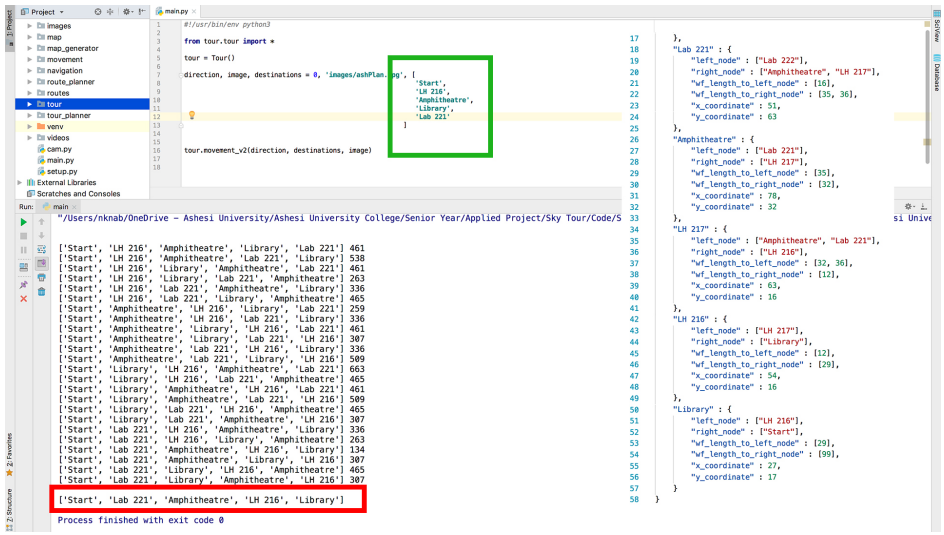

Figure 5.3 Test of tour planner functionality

The green rectangle in Figure 5.3 highlights the array of locations the user wishes to visit. The algorithm received an input of length 4 hence it was expected to produce 24 permutations. From the results obtained (Figure 5.3), we got 24 permutations and to their left was the cost gotten from the Dijkstra's shortest path algorithm. The least tour path has a cost of 108 which is in the red rectangle in Figure 5.3.

Based on the input image and the output image obtained from this functionality, it can be concluded that the tour planner functionality is valid.

Chapter 6: Conclusion and Recommendations

6.1 Summary

The aim of this project was to develop a navigation module for the drone tour guide. From chapter 2, the core features of the system were to perform a campus tour and some analytics drawn from the tours. The scope of this project ensures the campus tour's main navigation module functionalities were built and are fully functional. Unfortunately, the module could not be tested on a physical outdoor drone. This challenge will be discussed in section 6.2 of this chapter.

6.2 Limitations and Challenges

The major setback for this project was the inability to try the navigation module on a physical outdoor drone. This challenge was due to the fact that the drone was irresponsive. It could not take any sort of instruction both via code or the radio control receiver. Since the Erle-Copter is relatively a new programmable drone on the market, it was quite difficult to obtain enough help and resources to make the physical outdoor drone work. However, an attempt was made to try the navigation module on a physical indoor drone (CoDrone, an indoor programmable drone.). The feedback gotten from this experiment was positive however, the wind current kept drifting it of course because the physical indoor drone cannot withstand wind current.

6.3 Future Work

Testing on Physical Drone

In subsequent versions of this project, it would be great to see the implemented navigation and tour module running on an actual drone.

Adding other locations in Ashesi University College

With current phase of the navigation module, it only deals with the location on the Forecourt of Ashesi University College. Including the rest of the location at Ashesi University College would aid in giving complete tours.

Implementing a Camera Module

From the interview with Miss. Zeina Kowalski, she raised a point of having live virtual tours for prospective students and stakeholders. Therefore, the implementation of a camera module would achieve this goal.

Developing a User Interface

To aid with usability and also expand the functionality of the Sky-Tour System, a well-developed user-friendly interface will be appropriate.

Integration of Turtlebot II

Due to the availability of the Turtlebot II and the prior work done on those, it would great to have the Turtlebot II integrated with the Sky-Tour System. With this integration, there would be a relatively large amount of robot tour guides on the Ashesi University Campus.

Developing a Drone and Turtlebot II Ecosystem

Drones, as we know, have short flying times and this would cause problems such as drones crash landing or cutting tours short. To help solve this problem, an ecosystem made of drones and Turtlebots II will be useful. A proposed solution is developing a system where

drones can hand off the visitors they have to another drone or Turtlebot II to complete the tours.

6.4 Conclusion

Despite the challenges faced in the development of this module, this project was able to meet its main goal of developing a navigation and a tour planning module. However, this is not a complete product for the market. Although this is just a module of an application, it is a great foundation for future developers to build upon to have a robust application.

References

Acutronic Robotics. (n.d.). Robots: Erle-Copter. Retrieved from http://docs.erlerobotics.com/erle_robots/erle_copter

Acutronic Robotics. (n.d.). ROS: Basic Concepts. Retrieved from http://docs.erlerobotics.com/robot_operating_system/ros/basic_concepts

ArduPilot. (2016). Retrieved from http://ardupilot.org/

Boohene, K. (2017). Automated Collection and Visualization of Road Quality Data to Aid Driver Navigation.

Burgard, W., Cremers, A. B., Fox, D., Hahnel, D., Lakemeyery, G., Schulz, D., . . . Thrunz, S. (1998). The Interactive Museum Tour-Guide Robot. American Association for Artificial Intelligence. Madison.

Cober, D., Styler, A., & Naaktgeboren, A. (2006). Introduction to Robotics. Retrieved from https://www.cs.cmu.edu/afs/cs.cmu.edu/academic/class/16311/www/s07/labs/lab05/

Faber, F., Bennewitz, M., & Eppner, C. (2009). The humanoid museum tour guide Robotinho. Robot and Human Interactive Communication. IEEE.

Ivanov, S., Webster, C., & Berezina, K. (2017). Adoption of robots and service automation by tourism and hospitality companies. Revista Tur smo & Desenvolvimento, 1501 - 1517.

MIT Senseable City Lab. (n.d.). Retrieved from http://senseable.mit.edu/skycall/

Open Source Robotics Foundation. (2014). Retrieved from http://gazebosim.org/

Open Source Robotics Foundation. (n.d.). About Us: ROS. Retrieved February 2018, from ROS Website: http://www.ros.org/about-ros/

QGroundControl. (n.d.). Retrieved from http://qgroundcontrol.org/mavlink/start

QGroundControl – Drone Control. (2017). Retrieved from http://qgroundcontrol.com/

Robolink. (n.d.). Retrieved from https://www.robolink.com/codrone/

Sowah, N. A., & Afram, K. O. (n.d.). Retrieved from https://ashesirobotics.wordpress.com/2012/03/29/task4-path-planning/

Thrun, S., Bennewitz, M., Burgard, W., Cremers, A. B., Dellaert, F., Fox, D., . . . Schulz, D. (1999). MINERVA: A Second-Generation Museum Tour-Guide Robot. International Conference on Robotics and Automation, 3.

Yebisi, S. P. (2016). Tour Guide Robot.