ASHESI UNIVERSITY COLLEGE

TOUR GUIDE ROBOT

APPLIED PROJECT

B.Sc. (Computer Science)

Sheamus Punch Yebisi

2016

ASHESI UNIVERSITY COLLEGE

ASHESI UNIVERSITY COLLEGE

Tour Guide Robot

APPLIED PROJECT

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

Sheamus Punch Yebisi

April 2016

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

This applied project would not have been possible if not for several individuals in my life at this period so I acknowledge them.

Firstly, my gratitude goes to my supervisor, Dr. Ayorkor Korsah whose assistance and guidance helped me on my journey through this project.

Secondly, I would like to acknowledge all the lecturers that I have encountered in my time at Ashesi University College, for being there to guide me throughout this period of academic growth.

Thirdly, I would like to thank my family and friends for providing the support I needed in the times that I wanted to give up.

Finally, I would like give great honour to the Most High God who put all these people in my life. Thank you for giving me comfort when I needed it.

Abstract

Ashesi University College is periodically visited by individuals who are interested in seeing the campus. On their arrival, the university has a system in place to give an escorted tour to the visitor if needed. This project stems from the need for an ever-available tour system. The project seeks to replace the existing tour system, which consists of humans, to one that consist of robots.

The focus of this project is to map sections of the university campus, create a navigation framework and build a tour guide application that uses the navigation framework to provide visitors with an escorted tour of the Ashesi University campus.

This paper presents a detailed description of the implementation of a tour guide robot and its limitations. Finally, the paper outlines short comings of the project as well as future work for the project to improve the tour guide robot.

Table of Contents

List of Figures

Chapter 1: Introduction

1.1 Aim

This applied project aims at designing and implementing a robotic tour guide that is able to lead visitors around Ashesi University College's campus and provide them with descriptions of the places they see. With such a system, many people can be taken for a tour in rapid succession with a robot that has fewer needs than a human tour guide, who would need to eat, who needs to take bathroom breaks and who has other needs of a human. This project focuses on building a tour guide system that is capable of efficiently navigating the campus, is able to effectively communicate with visitors and is understandable enough for a visitor who has no experience at all operating the tour guide to control it with ease.

1.2 Background

Ashesi University College is visited by many people due to its reputation for change in Africa. For the university to be able to broadcast its vision – an African Renaissance driven by a new generation of ethical entrepreneurial leaders – it is necessary for people to visit and see how Ashesi hopes to cause this change. The university hosts many prominent figures who arrive to see what is becoming of the vision of Ashesi's founder, Patrick Awuah; many of the university's investors, who pass by to see their investment; and, most importantly, families of aspiring students. From the constant flow of visitors to Ashesi arises the need for someone to be constantly available to take care of any visitors. That is, on any occasion that visitors arrive on the campus, there must be someone assigned to them who will lead them around and provide them with information about the areas seen.

The university has a tour ambassador system that consists of student volunteers who are responsible for taking visitors around the campus. The university's campus is not as large as that of many public universities in Ghana, but it is still tiresome walking from one

end of the campus to the other. Another feature of the tours led by the guides is their repetitive nature. The tours involve the tour guide moving from one location to another and giving a description for each. Although the ambassadors find this an enjoyable task, it is possible to reduce their responsibility. One of the ways to reduce the ambassadors' responsibility is the provision of an automated system to replace them. That is what this project hopes to do by providing a tour guide robot.

1.3 Related Work

One other place where many people visit is a museum. Visitors come from all around the world to see its exhibits. The Deutsches Museum in Germany is one that has integrated a tour guide robot into its establishment. One of its projects involved a robot called RHINO, which was programmed to be able to take visitors around to view the museum's exhibits.

Using an on-board screen, visitors were able to interact with the robot in ways such as, knowing where the robot was going and getting more information about an exhibit during the tour. RHINO also had recordings that were played when it reached an exhibit on the tour, to give the visitors a description of the exhibit. (Burgard, et al., 1998). These were the two main ways that RHINO interacted with the visitors of the museum, other than the horn it used to alert people of its presence. The RHINO's large cylindrical mass seemed to stand out against the usual items found in a museum, which majorly characterized it.

Another robot comparable to RHINO is MINERVA that was installed in the Smithsonian's National University of American History. MINERVA is a generation after the RHINO with developmental improvement such as learning of maps, the use of a ceiling mosaic to find where it was, an improved path planner, an automated tour composition, an enhanced interactive interface and an improved Web interface (Thrun, et al., 1999). In comparison to RHINO MINERVA was more efficient in communicating with people. Apart

from simply leading the visitors around, its improved on-board interface, which was a face, attracted people to it (Thrun, et al., 1999). The ability to express humanlike gestures is one of the major determinants of whether a robot is a successful tour guide because it connects with a visitor.

Other robots such as Robotinho, another robot that was tested in the Deutsches Museum, have been given a humanoid look in order to make interaction between the robot and the human more natural (Faber, et al., 2009). Interaction between the tour guide, in this case a robot, and the human being is very essential as it increases appreciation of a tour for people who are very likely to lose focus during a tour. The humanoid robot incorporated gestures that are used by humans, to allow the visitors to better interpret what the robot was referring to and meant. (Faber, et al., 2009).

At the centre of these three cases, the robots were required to move in a given environment, by manoeuvring around obstacles, in order to carry out their tours successfully. To do this they all used an effective navigation module that would allow them to overcome the problem of moving into the right position and interacting with the visitors as an escort tour guide.

1.4 Project History

This applied project continues an on-going project that was initiated to create a tour guide robot for the Ashesi University College community. As such, this section will discuss the progress of this project prior to the start of this applied project.

1.4.1 Tools

Two main tools were used in the development of the tour guide robot prior to the start of this project. These were a Turtlebot and an Android tablet which acted as the user interface.

Turtlebot. This is a programmable personal robot. It runs a Robot Operating System (ROS) that makes programming the robot easier. ROS will be discussed in Chapter 3 in more detail.

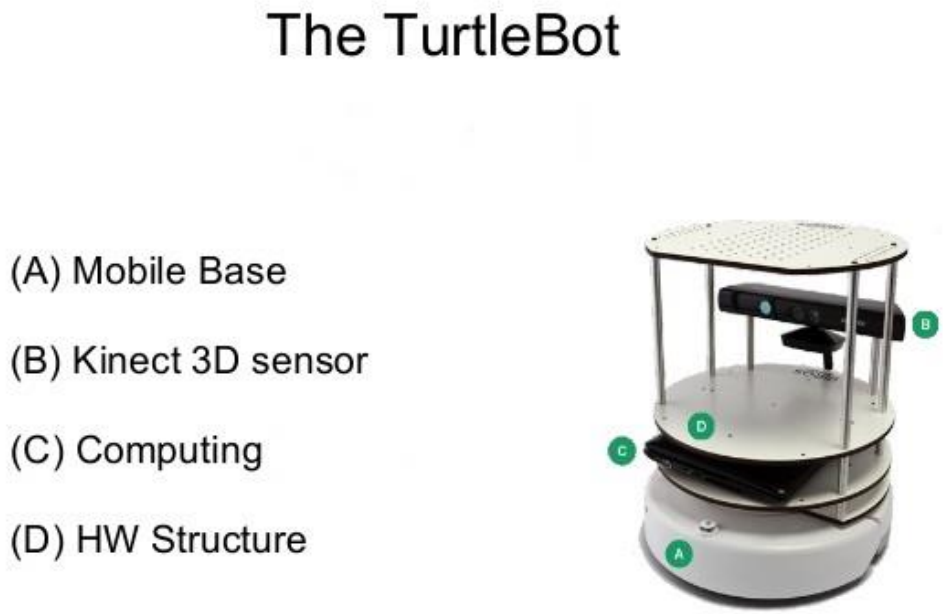

Figure 1.1 Labelled Turtlebot

Handheld device. This is a mobile tablet running an Android operating system which acts as the interface for communicating with the Turtlebot.

Figure 1.2 Android tablet

1.4.2 Description

The tour guide robot constitutes a Turtlebot robot and a tablet. The Turtlebot is the robot that takes a visitor around the campus while the tablet acts as the user interface which is used to interact with the robot. A tablet is chosen to be the user interface because the Turtlebot is small and so to prevent the user from having to bend over the Turtlebot to interact with it, a handheld device was choosen.

1.4.3 Implementation status

Below is a summary on the progress of the work prior to the start of this applied project.

- Given a file containing a list of locations, the tour guide robot was able to move from one location to the other starting from the first and ending at the last.
- A concept of the user interface for the tour guide robot was created. In this concept the interface sent one number from a list, representing the interested locations, to the tour guide robot every time a button was pressed.

- The tour guide was able to play recordings of the descriptions of locations.

1.5 Objectives

This section outlines the objectives for this applied project.

1.5.1 Mapping

Prior to the start of this project, there existed a collection of maps representing a limited number of areas in the university. This limited number of maps reduces the places where the robot can go. This project hopes to increase the number of maps to include the main block of Ashesi University College.

1.5.2 Route planning

In the initial implementation of the tour guide robot, routes are not planned to find the most efficient way of travelling to a set of locations. The arrangement of the locations in the file used for movement is the way in which locations are visited. This applied project will enable the tour guide robot to create its own route through locations by taking into consideration ‘cost’ of moving between locations.

1.5.3 Movement

In the initial implementation of the tour guide robot, the tour guide robot was unable move freely between different maps. This was due to a design decision, which will be discussed in Chapter 4, made by the researchers that worked on the project. This project hopes enable free movement between maps.

1.5.4 Application Program Interface for a reusable navigation module

This project hopes to provide a navigation Application Program Interface (API) that can be adapted to other projects concerning navigation on the Ashesi University campus. This will make it simple for future developers to move the robot anywhere on the campus.

1.5.5 Tour guide application

This project hopes to create a tour guide application that is able to escort individuals around the Ashesi University College campus and provide them with description of the places visited.

Chapter 2: Requirement Analysis

2.1 User Requirements

2.1.1 Approach

In order to gather the requirements for this project, the stakeholders in the project were defined. The main stakeholders in this project were the users of the tour guide robot and hence they needed to be contacted. Other stakeholders in the project were the researchers that had started the project so to gain technical understanding of the project they were contacted. Individuals in user groups were sampled at random and interviewed. The insights gained from the interviews can be found in Appendix A.

2.1.2 Scenarios

Based on the insights drawn from the interviews with stakeholders in the project, scenarios were formulated in order to simulate the function of the tour guide robot in cases that may arise. These scenarios will be used in finding the use cases of the project.

Scenario 1. Madonna a banker and mother of a SHS student, arrives at Ashesi University College. She has been told about the university by her colleague at work and would like to see for herself what the school has to offer her child. Upon arrival, she goes to the reception and meets Derrick, the receptionist, who directs her to the office in charge of tours. Darlene, who is on duty for the day, chats with Madonna for a while and understand that she want a tour of the school. Darlene takes the tour guide robot outside and positions it appropriately. On the tour guide robot's user interface she then clicks "start tour" and hands it over to Madonna to start her tour. Madonna follows the robot tour guide as it leads her around the campus, stopping at points of interest and giving a description of each one. When the tour is over, the tour guide returns to Darlene and says goodbye to Madonna. Madonna returns the user interface to Darlene, who asks how the tour went. Thereafter,

Madonna leaves the school feeling accomplished, happy with her new knowledge of the university.

Scenario 2. On the way to Accra from Aburi, Joe, a busy trader happens to be driving on the Berekuso road–the road in front of Ashesi University College–when he sees the Ashesi sign and decides to pass by and see what kind of university was situated on the outskirts of town. Upon arrival, he goes to the receptionist who directs him to Jacob, the personnel in charge of tours that day. Joe is in a hurry and hence informs Jacob that he only wants to see the labs that the campus has. Jacob takes the tour guide robot to a starting point and positions it. He then launches the user interface on a tablet and chooses the places that needed to be visited. Having setup the custom tour he hands the user interface to Joe. The tour guide move straight to the first lab and then on to the next, giving a detailed description of each one when it arrives in position. After the last lab has been visited, the tour guide leads Joe back to the starting location, where they are met by Jacob who asks Joe about his tour. Joe is then able to leave quickly to continue his journey into town.

2.1.3 Use Case Diagram

This section uses the insights from the scenarios to generate a graphical representation of each users' usage of the application. By doing this measurement of the project's completion can be measured.

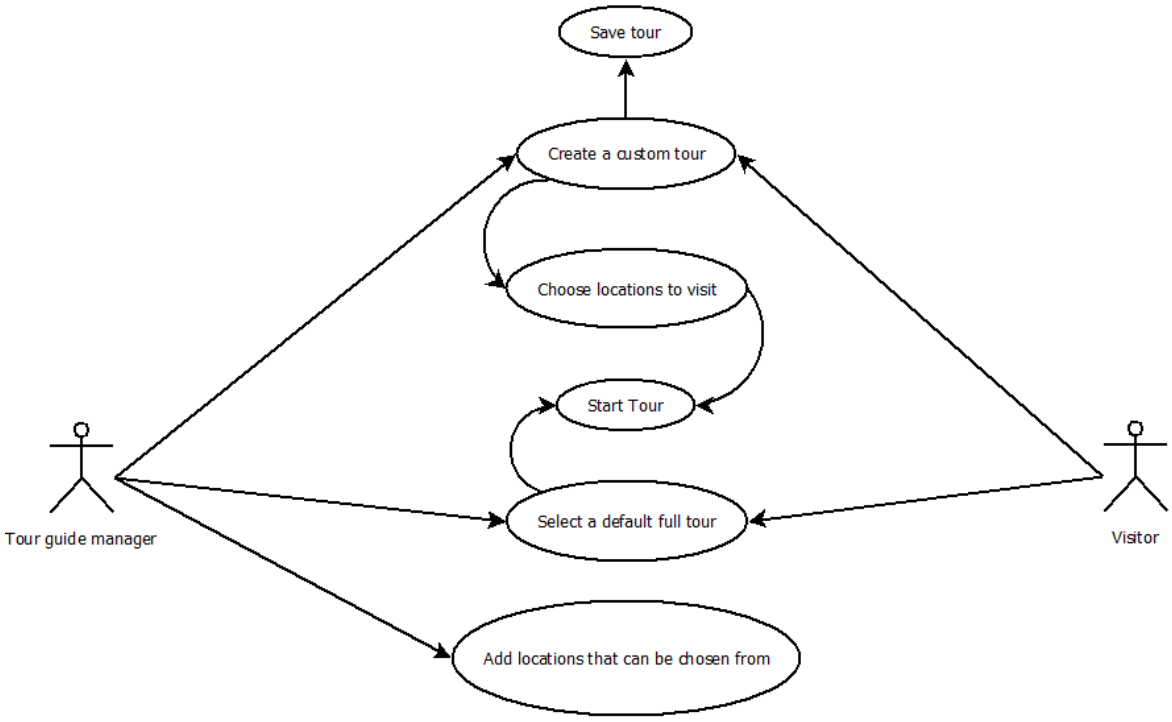

Figure 2.1 Use case diagram

2.1.4 Functional requirements

This section presents a list of what the application should be able to do.

1. The user can choose where she/he would like to visit. Not all users are likely to want, or have the time to, visit all the locations on the university's campus; as such, a user should be able to choose the places where she/he wants to go.
2. The user can select a default tour. This will be available to users who are interested in taking a predefined tour. These default tours will consist of a set of locations that a user will go to on the tour. This will save the user's time from going through all the locations and selecting them individually.
3. The user can start a tour. When a user starts a tour, she/he will be taken around the university's campus and shown the various locations that are of interest on the tour.

2.1.5 Non-Functional Requirements

2.1.5.1 Product

1. The system should be able to generate the shortest tour path for a user. Given the set of locations a user would like to visit, the robot must be able to lead her/him along the shortest route. If this is not done the robot could lead the visitor in circles, which would waste the visitor's time and also the battery of the robot.

2.1.5.2 Organizational

1. The system should not lead visitors into dangerous or restricted areas. The tour guide robot should not be able to access areas that are out of bounds.

2.1.5.3 External

1. The system should be secure against theft. This suggests addition of security to prevent loss of the robot. This will be essential because the robot is highly portable and can be carried away by a user.

2.2 System Requirements

This section defines what the system must have in order to meet the functional requirements defined. The section is broken into user interface and tour guide robot to cater for the two parts of the system.

2.2.1 User interface

1. There must be a screen that allows a user to select locations to visit on a tour from a list.
2. There must be a screen that allows a user to select a tour.
3. There must be a screen that allows a user the start a tour.
4. There must be a screen that allows a user to save a tour. This tour would consist of the selected locations.
5. There must be a screen that allows a user to add a new location.

2.2.2 Tour guide robot

1. The Turtlebot should be able to move from one point to another point. The purpose of a tour guide is to show people around, which involves movement from one place to another. Hence, the robot must be able to accomplish this task and be able to go to specified locations.
2. The Turtlebot should be able to give some information on a particular location to the user. The robot needs to be able to play a recording to the user when it reaches a particular location. The recording must contain relevant information with respect to the location. Since the tour guide robot is a substitute for the human it must be interesting in order to keep the attention of the visitor.

Chapter 3: Architecture and Design

3.1 Project Overview

The end goal of this project is to create a tour guide robot that will be able to take visitors around the Ashesi University College campus. This tour guide robot will lead a visitor around, visiting different locations on campus and giving the visitor an audio description of the location.

3.2 Hardware

Before discussing the architecture, this section will provide an overview of the hardware components involved in this project. There are three main hardware components namely, a Turtlebot, a workstation server and a handheld mobile device running Android OS. As such, the project is structured as a distributed system. The Turtlebot has already been discussed in the previous chapters. The workstation server will be the computational hub for the project, that is, all the project's logic will be located on it. A handheld Android device will be used as the user interface for the tour guide robot. The reason for using a handheld device is attributed to the low height of the Turtlebot which cancelled the possibility of an on-board interface which users will struggle to interact with.

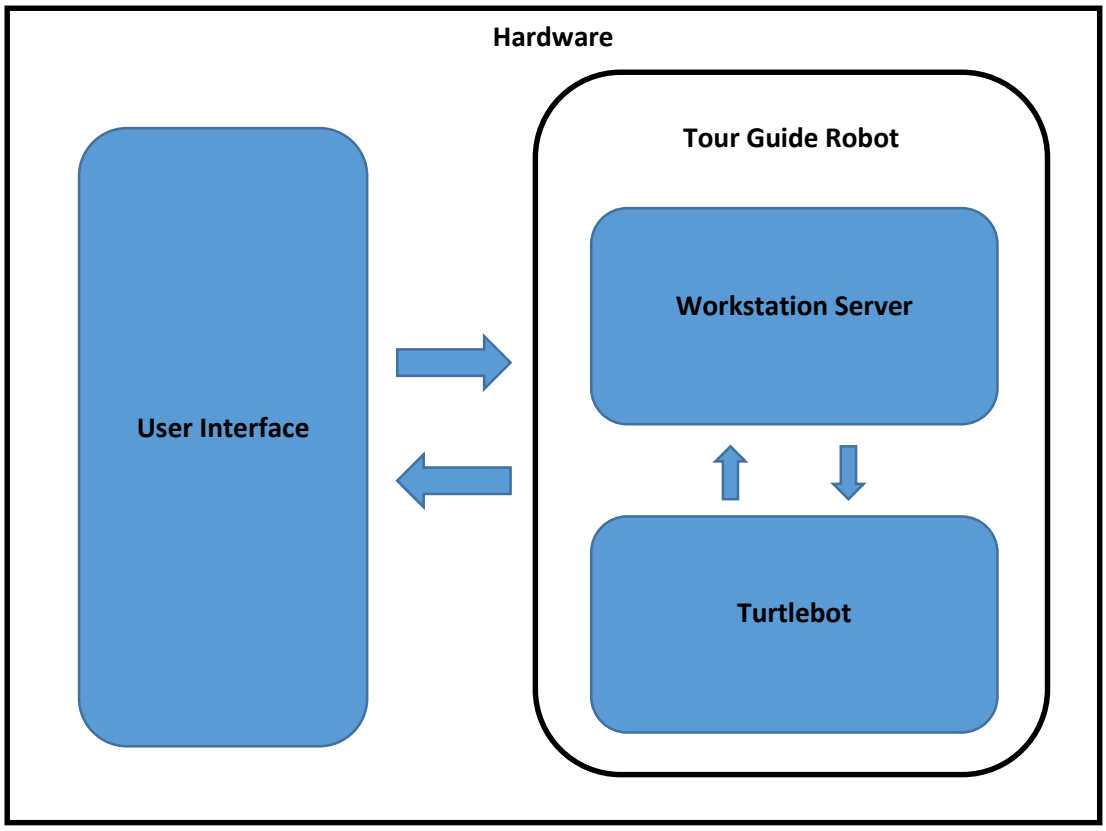

Figure 3.1 Interaction between the hardware components

3.3 Robot Operating System

Robot Operating System (ROS) is the framework that will be used in developing the tour guide robot and therefore it will majorly define the design of the architecture. Hence this section gives an overview of ROS. According to the ROS.org website:

> ROS is an open-source, meta-operating system for your robot. It provides the services you would expect from an operating system, including hardware abstraction, low-level device control, implementation of commonly-used functionality, message-passing between processes, and package management. (Thomas, 2014)

ROS is used in robot development making it simpler to program a robot to complete a particular action due to the abstraction of the functionality. ROS allows developers to create what are known as ‘nodes’ which are largely independent programs (O'Kane, 2013). Some nodes may provide services that other nodes can request for to use. Communication in ROS is optimized for distributed systems, such as is found in the tour guide robot which comprises a workstation and the Turtlebot. Nodes may ‘publish’ what are called topic so that any other node can ‘subscribe’ to the topic and use the service.

3.4 Project Modules

In order to accomplish this task, there are different modules constructed to perform different tasks. The modules are as follows:

- User Interface module
- Interface Relay module
- Tour Control module
- Route Planning module
- Movement module
- Map Loader module
- Sound module

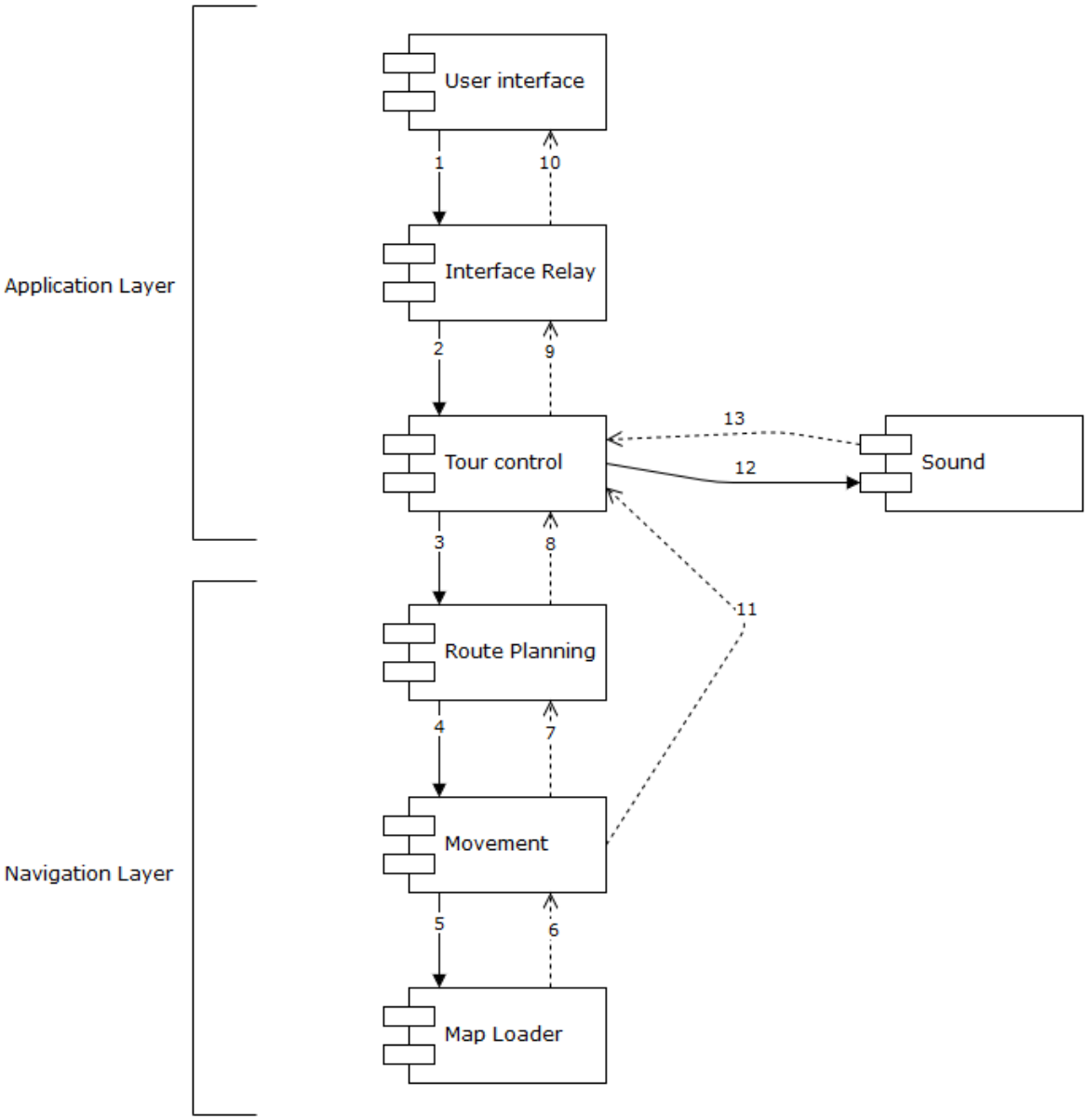

Figure 3.2 Diagram showing the interaction between the project modules

3.4.1 User Interface module

This module is an Android application that allows a user to select the locations that she/he wants to visit and also start a tour. This module contains the layer that directly interacts with a user. The Model-View-Controller design principle will be used in implementing this module.

3.4.1.1 Model

This will contain all scripts that describe the data that will be used in the project.

Models

1. Location: The location model describes all the attributes that need to be recorded concerning a point on the campus. These attributes include:
    a. Name: This is the full name of the location e.g. Library, Lab 222. The name will enable a user to easily identify where she/he might want to explore.
    b. ID: This identifier will be what is sent to the workstation server to allow a route to be planned through given locations.
    c. Description: This will be used to give the user more information about a location.
    d. Image: A link to an image of the location.
2. Tour: The tour model describes all the attributes that need to be recorded concerning a tour. These attributes include:
    a. Name: This is the name given to a tour.
    b. Locations: This is the set of locations that are to be visited in the tour.
3. Adapter: The adapter model will define the look and feel of locations and tours viewed on the screen by a user.

3.4.1.2 View

This will contain all the scripts pertaining to the look of the user interface for the Android application.

Views

1. Home screen: On this screen a user will be asked to select between two options which are, “select a tour” and “create a tour”. Selecting a tour will allow a user to

select a predefined tour to start. On the other hand, creating a tour will enable a user to make her/ his own tour by selecting the locations that should be visited.

2. Create tour screen: On this screen a user will be allowed to select from a list of locations that the tour guide robot should visit.
3. Select tour screen: On this screen a user will be allowed to select a tour from a list and start it.
4. Preference Screen: On this screen a user will be able to change the tour guide robot that she/he is communicating with.

   Inputs

   - IP address of the communication channel.
   - Port for communication.
5. Add dialog: On this dialog a user will be allowed to add a location that can be visited to the selection screen.

   Inputs

   - Name of location
   - Description for location
   - ID of location
6. Tour Screen: On this screen the progress of the tour will be shown to the user.

   Outputs

   - Image of the current location.

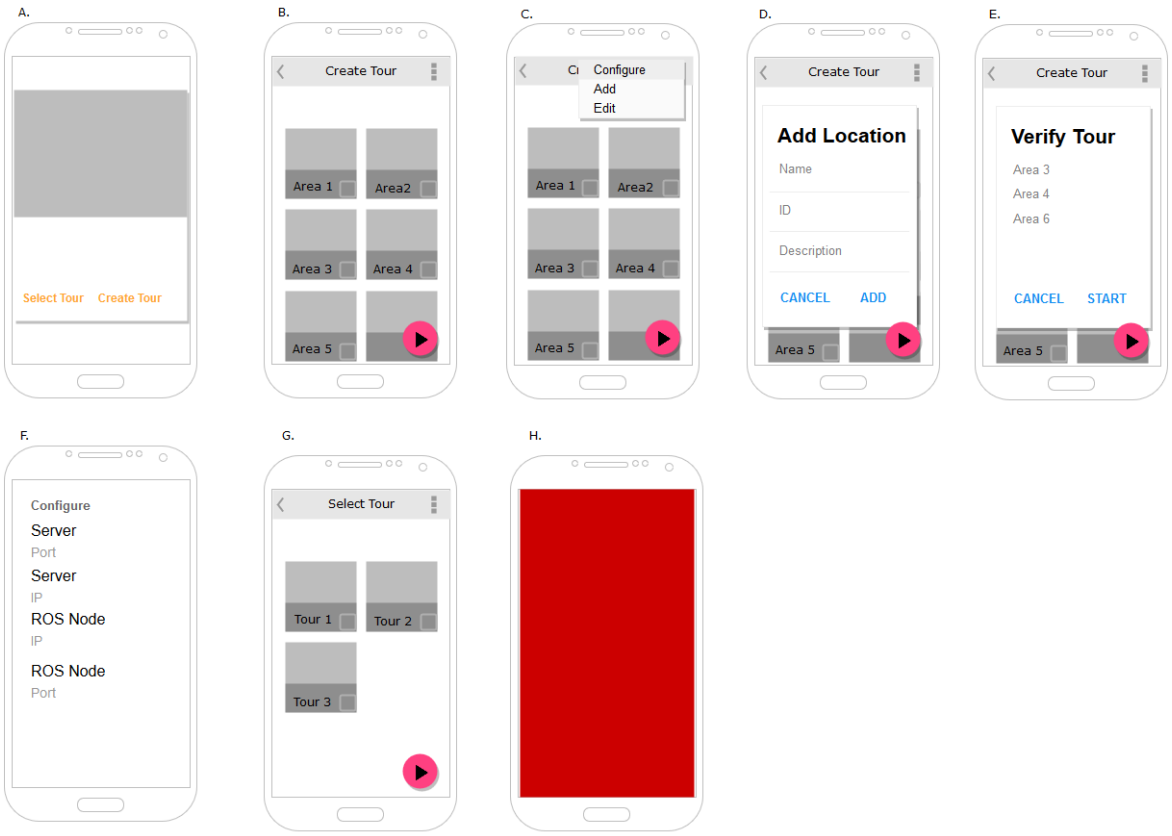

Figure 3.3 Mock-up design for the screen of the user interface

A – Shows the home screen.

B – Shows the create tour screen.

C – Shows the configuration options.

D – Shows the Add location dialog.

E – Shows a confirmation dialog.

F – Shows the settings screen.

G – Shows the select tour screen.

H – Shows the tour progress screen.

3.4.1.3 Controller

This will contain all the scripts that involve interaction with the workstation server and storage control in the application.

Controllers

1. Communication service: This is a service that will enable communication between the user interface and the robot tour guide.
2. Storage manager: This will control how data for the user interface is stored.
3. Selected location store: This will hold the locations that have been selected for a tour.

3.4.2 Interface Relay Module

All data communicated between the tour guide robot and the user interface will be passed through the interface relay module. It will provide the channel of communication via a socket connection. A socket is used to easily facilitate communication between the two platforms, ROS and Android.

The roles of the interface relay module include:

- Receiving locations that are to be visited, and passing them to the tour control module.
- Receiving notifications on the tour's progress and sending them to the user interface.

3.4.3 Tour Control Module

This module handles all the logic concerning the tour application. It uses the navigation layer (Route planning module, Movement module and Map loader module) to move the robot and receives updates which it uses to decide whether to move on or play a sound for the location reached.

3.4.4 Route Planning Module

This module will contain the implementation of a route planning algorithm, which will allow the tour guide robot to move around the university campus through specified locations, using the available maps. In order to do this a set of location points must be provided as well as a start point. To solve this route planning problem it was modelled as a Travelling Salesman Problem (TSP) and a TSP solution approach was applied to solve it. TSP describes a problem where, given a set of node one must find the shortest route through all of them and back to the start, going through each node only once (Cormen, Leiserson, Rivest, & Stein, 2009). Hence, the route planning module would apply TSP to find an optimum route.

Inputs to route planning module

- Set of locations to visit.

Outputs of route planning module

- A sequence of point to traverse.
- A set of locations to visit.

3.4.5 Movement Module

The movement module will control how the tour guide robot moves from point to point over the course of the tour. This module will be updated from earlier work. Below are the functions of this module:

- Provides the link of a map to the map loader module.
- Provides the functionality to enable the robot to know where it is in a map.
- Moves robot from point to point within the map.
- Offers a service to receive the set of points to be moved to.

- Provide notification that destination has been reached.

Inputs to movement module

- A sequence of points to visit.

Outputs of movement module

- A link to a map file. This is periodically sent to the map loader when the current map needs to be changed.
- Notification that certain points have been reached.

3.4.6 Map Loader Module

The basic functions of this module are:

- Receiving a link of a map from the movement module.
- Changing the map that is being moved through.

3.4.7 Sound Module

The basic functions of this module are:

- Receiving a link of a sound from the tour control module.
- Playing a sound.
- Returning the length of a sound to the tour control module via a ROS service to indicate how long to wait before requesting movement to resume.

3.5 Inter-Module Communication

In Figure 3.2, communication between the modules has been represented by numbered directed edges. The communication is divided into messages, indicated by solid lines, and responses, indicated by dashed lines. Following are the description of each of the lines of communication.

3.5.1 Messages

Message 1

This represents communication between the user interface module and the interface relay module. The purpose of this communication is to pass the locations the user would like to visit to the tour guide robot. This is done by use of a socket connection between the two modules.

Message Structure

This is a JavaScript Object Notation (JSON) Array with the following form:

[<point 1>, <point 2>…<point n>]

Message 2

This represents the communication between the interface relay module and the tour control module. The purpose of this communication is to pass the data retrieved from the user interface module to the tour control module in order to initiate the tour process. This is done by subscription to a ROS service that is offered by the tour control module to collect the data.

Message Structure

This is a JSON Array with the following form:

[<point 1>, <point 2>…<point n>]

Message 3

This represents the communication between the tour control module and the route-planning module. The purpose of this communication is to pass the data retrieved from the user interface module to the route-planning module in order to construct a tour route. This

will be done by subscription to a ROS service that is offered by the route-planning module to collect the data.

Message Structure

This is a JSON Array with the following form:

[<point 1>, <point 2>…<point n>]

Message 4

This represents the communication between the route planning module and the movement module. The purpose of this communication is to pass the sequence of points that need to be travelled by the tour guide. This will be achieved by subscription to a ROS service offered by the movement module, which requires the sequences of points.

Message Structure

This is a JSON Array with the following form:

[<point 1>, <point 2>…<point n>]

Message 5

This represents the communication between the movement module and the map loader module. The purpose of this communication is to pass the path of a map that needs to be loaded. The map-loading module offers the service that allows the map path to be passed to it.

Message Structure

/home/turtlebot/Desktop/the_tour_guide_maps/<map name>

Message 12

This represents the communication between the tour control and the sound module. The purpose of this communication is to send the path to a sound that needs to be played. The service to receive the link to the sound is offered by the sound module.

Message Structure

/home/turtlebot/Desktop/the_tour_guide_sounds/<sound name>

3.5.2 Responses

Response 6

This represents the response from the map loader module to the movement module. This response is a basic “SUCCEEDED” or “FAILED” report for the movement module to know what to do next.

Response 7

This represents the response from the movement module to the route-planning module to acknowledge receipt of the sequence of points.

Response 8

This represents the response from the route-planning module to the tour control to acknowledge receipt of the set of interest points.

Response 9

This represents the communication between the tour control module and the interface relay module. This communication will be used to acknowledge receipt of the set of locations. It will also give progress notifications that need to be relayed to the user interface.

Response 10

This represents the communication between the interface relay module and the user interface module. This communication will be used to acknowledge receipt of the set of locations. It will also by means of a socket connection relay the progress of the tour to the user interface module.

Response 11

This represents the communication between the movement module and the tour control module. This communication channel will be used by the movement module to notify the tour control module that it has reached a particular location.

Response 13

This represents the response from the sound module to the tour control module. This response notifies the tour control module that the sound that was requested to be played is playing. The response is also used to communicate the duration of the sound.

3.6 Map Metadata Storage JSON

In order for the route planning module to do its work, which is to create a route that the tour guide should take and send this information to the movement module to complete the tour, the maps' meta-data is needed. This meta-data will be stored in a JSON Object. This object has been optimized to decrease search time for elements within it, which is the main reason why it was chosen over its XML predecessor. The structure of the object holding the maps' meta-data is shown in Figure 3.4.

```
{    "maps": {
        "<map name>": {
            "link": "<link to map>",
            "points": {
                "<interest point>": {
                    "overlap": false,
                    "distance": {
                        "<interest point 1>": <distance>,
                        "<overlap>": <distance>
                    },
                    "sound": "<link to sound>",
                    "x": <x-coordinate>,
                    "y": <y-coordinate>,
                    "z": <z-coordinate>,
                    "ox": <x-orientation>,
                    "oy": <y-orientation>,
                    "oz": <z-orientation>,
                    "ow": <w-orientation>,
                    "next_map": ""
                },
                "<overlap>": {
                    "overlap": true,
                    "distance": {
                        "<interest point 1>": <distance>,
                        "<interest point 2>": <distance>
                    },
                    "sound": "",
                    "x": <x-coordinate>,
                    "y": <y-coordinate>,
                    "z": <z-coordinate>,
                    "ox": <x-orientation>,
                    "oy": <y-orientation>,
                    "oz": <z-orientation>,
                    "ow": <w-orientation>,
                    "next_map": <mapname>}}}}}
```

Figure 3.4 Structure of JSON storing the metadata of the maps

In the structure seen above there is a root container called **maps** which holds objects of the individual maps. The **map name** is used as a reference to its attributes. Each of the maps have two main attributes, the **link to the map** and its **points** which refers to the locations in the map. As with the maps, each point references its attributes. Each point has 11 main attributes, an indicator, specifying whether it is an **overlap** or not; **distances**, which contains the distance from the point in focus to the other points in the map; **sound**, which is a link to the sound for the location; coordinates, **x**, **y** and **z**; orientation, **ox**, **oy**, **oz** and **ow**; and the **next map**, which will be available if the point is an overlap.

Chapter 4: Implementation

4.1 Implementation Concepts and Tools

This section describes concepts and tools that will be used in the implementation of the tour guide robot.

4.1.1 Localization

This is the process whereby the robot tries to determine where it is in a map (Mataric, 2007). This is necessary because a robot does not know where it is in a given environment. However, in order to accomplish any task in an environment, the robot must first know its position and its orientation. In this project, localization is initiated by the provision of the coordinates, which represent the robots position and orientation in a map, to the robot. The coordinates are obtained upon request from the JSON structure where they are stored.

To setup this sort of localization the amcl (Adaptive Monte Carlo Localization) package that is provided by ROS will be used. This amcl package facilitates localization by means of the Monte Carlo Localization technique (NilsBerg, 2015). This is a probabilistic method of localization, which uses random samples indicating probability density. The system chooses from the set of samples and then weighs them checking the likelihood that the robot is actually at that point, thereby thinning out the possibilities and leaving only the most likely probable location of the robot (Dellaert, Fox, Burgard, & Thrun, 1999). This is a relatively quick and memory efficient method of localization and is what is used by the amcl package in ROS to keep the robot localized.

The amcl package subscribes to the topic called ‘initialpose’ from which it gets the robot’s starting position. This starting position, which consists of the pose, the position and orientation, and the covariance of the point, is what will enable the robot to localize initially.

The amcl system also subscribes to another topic called ‘map’ from which it will retrieve the map that it is currently in and in which localization is to be performed.

Without accomplishing localization inside of a map it will be impossible for the robot to move, which is needed for the tour guide to be able to show visitors around the school.

4.1.2 Movement

Movement describes how the robot gets from one point to another in a particular map. To accomplish movement in this project, another ROS abstraction, the navigation stack, which is on the robot, is used. This navigation stack contains packages that enable the robot to move successfully between two points. The following make up the ROS navigation stack:

- Transform
- Sensor Information
- Odometry Information
- Base Controller

Transform. This involves the configuration of the ‘tf’ package included in ROS. The ‘tf’ package keeps track of the multiple coordinate frames. It also maintains the relationship between the coordinate frames, which allows for transformation between coordinate frames. Hence, in order to be able to find the relationship between the frames, which is a necessity for movement, the ‘tf’ package must be configured.

Sensor Information. The data retrieved by the sensors on the robot is very necessary to navigation because it helps in the re-localizing process of the robot while it is on the move, as well as in obstacle avoidance. Therefore the robot must publish its sensor data to a topic so as to be used in the navigation process.

Odometry Information. As its name suggests, it deals with the publishing of the odometry data gained from the robot. Using odometry, a robot is able to keep track of its location in a map with respect to the starting position. By this definition it is evident why it is necessary for the odometry information to be included in the navigation stack.

Base Controller. The navigation stack expects velocity commands to be sent to a particular topic, but for these velocity commands to be of any use they must be captured, converted to motor commands and sent to another package that will interpret it and move the robot accordingly. Hence, the base controller performs this task of retrieving the velocity command, converting them into motor commands and then sending it to the mobile base, which will move the robot.

In order to use the navigation stack, the move base package in ROS is used. A goal is given to this in the form of a pose and orientation and then using the navigation stack it tries to reach the destination. It is important to note that all of this is done after localization.

4.1.3 Mapping

In order for the tour guide robot to initiate a tour, it needs to be given an environment. In this environment it will be able to navigate and hence lead visitors around to different areas. The creation of the environment will be done using the on-board sensors of the robot. The environment that will be created is represented as a Portal Gray Map (PGM) image, which is an image format that stores gray level values; the value of brightness. Applying a PGM format image in the creation of the environment 3 levels of brightness are used, black, gray and white which represent obstacle, unexplored and free respectively. Using this a 2dimensional representation of any environment can be constructed.

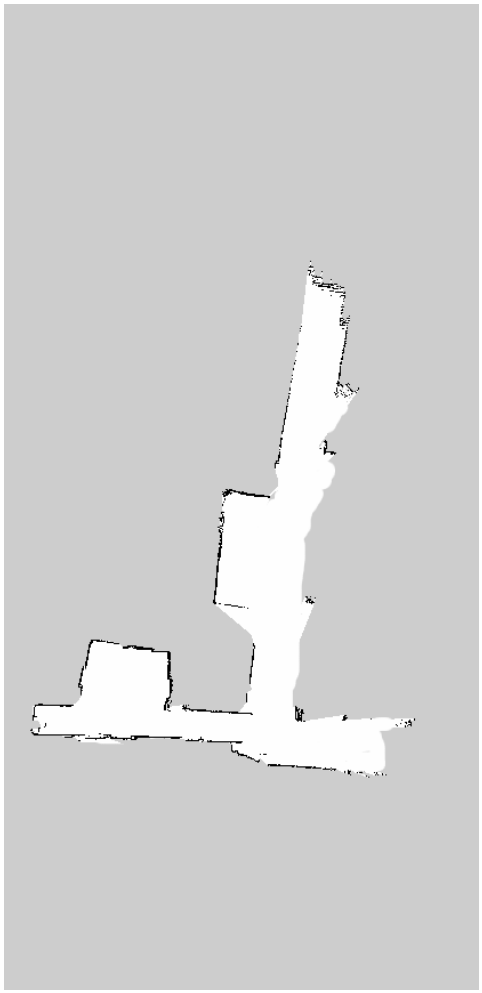

Figure 4.1 2Dimensional environment represented in a PGM image

In order to do this ROS provides a package called ‘gmapping’ which uses a laser based SLAM process to construct a 2D occupancy grid map. SLAM stands for simultaneous localization and mapping. The map constructed by this component is saved to be used at a later time by the movement component. In order to use the map at a later time the ‘map_server’, another ROS package, is used to load the map.

4.2 Prior Work

This section aims at discussing the work implemented prior to the start of this applied project. Listed below is the project’s progress prior to the beginning of this applied project.

4.2.1 Mapping

Maps for some sections of the campus were built and stored. These maps had meta-data which was stored in XML files and contained the coordinates for places that would be visited within the map. Also in these XML files were the regions where the map overlapped with another map. Every map had exactly two overlapping regions, an overlap for entry and an overlap for exit. This meant the tour guide robot would only be able to move linearly through maps. Figure 4 shows how navigation was accomplished between to overlapping maps, the green spot represents the entry overlap, red spot represents the exit overlap and the blue line indicates the path of the robot.

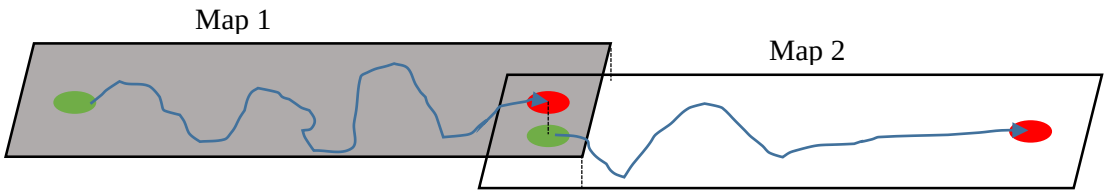

Figure 4.2 Image representing navigation between two overlapping maps.

4.2.2 Movement

There was a module that allowed the tour guide to move through the maps. It did this by retrieving a list of locations with attributes that stored location meta-data from an XML file. The XML files defined the route of movement, so in order to change the route the XML file needed to be changed. By means of an XML file the tour guide could be given a start location, which it would use to find itself in a map, localize, and move to the positions noted in a XML file. When overlap regions were reached, a new map would be loaded, as well as a XML file to guide the movement. A sample of an XML file used by the movement module is shown in Figure 4.3. In the figure the locations to be visited are represented as interest points.

```
k?xml version="1.0" encoding="UTF-8"?>
<map>
  <file_location location="/home/turtlebot/Desktop/maps/3.yaml" />
  <interest_point id="1" name="Bathroom" x="4.66738" y="-0.281095" z="0.0186685" w="0.999826"
spiel="/home/turtlebot/Desktop/spiels/1.m4a" />
  <interest_point id="2" name="Salmoe's Office" x="8.6262" y="0.338128" z="0.0186685" w="0.999826"
spiel="/home/turtlebot/Desktop/spiels/2.m4a" />
  <interest_point id="3" name="Dean K's Office" x="9.93577" y="0.863163" z="0.0186685" w="0.999826"
spiel="/home/turtlebot/Desktop/spiels/3.m4a" />
  <interest_point id="4" name="Kasper's Office area" x="16.8597" y="2.28825 " z="0.0186685" w="0.999826"
spiel="/home/turtlebot/Desktop/spiels/4.m4a" />
  <overlap id="1" next_map="END" exit_region="false" x="0.872382" y="0.147806" z="0.0186685" w="0.99624" />
  <overlap id="2" next_map="END" exit_region="true" x="17.4588" y="3.36634" z="0.0186685" w="0.999826" />
</map>
```

Figure 4.3 Sample of information found in the XML file

4.2.3 User interface

The development of an Android application whose purpose was to act as an interface for the tour guide robot was started. The application provided a means to start a tour and to relay the locations of interest to the tour guide robot. Prior to this project the application used a socket connection to send data from a static array, representing a list of locations, to the tour guide robot.

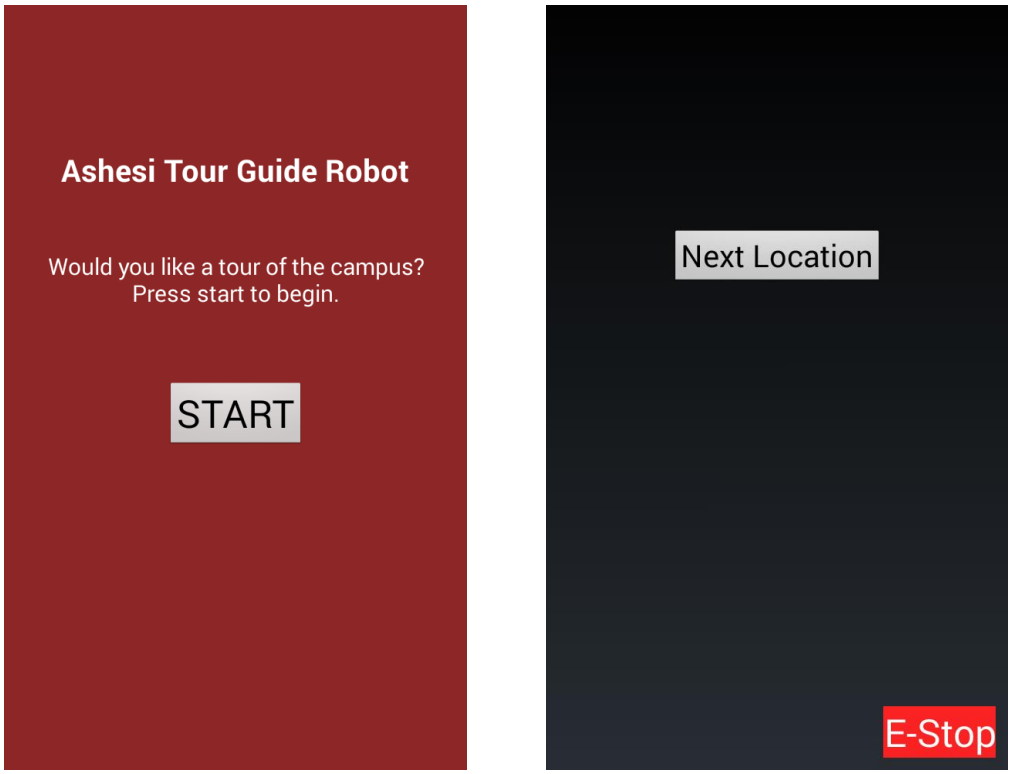

Figure 4.4 User interface prior to the start of the project

4.3 Implementation Process

4.3.1 Navigation Layer

This section describes the implementation process of a generic navigation layer, an abstraction new to the project, which is used to perform the route planning, the movement and the map loading tasks of the robot. Each of these modules are implemented as separate ROS nodes.

4.3.1.1 Route planning

The route planning module was developed on the workstation server of the tour guide robot as a single ROS node. It provides the service that receives a set of locations and uses a routing algorithm to find the most efficient way to go from the start to all the given locations. In prior work on the tour guide robot, there was no route planner, however XML files such as the one seen in Figure 4.3 were provided to the robot tour guide, which moved to the location in the order listed.

In the route planner, the optimum shortest route is found using a greedy approach of solving the TSP. This approach uses an algorithm that iteratively finds the closest location of interest from the location it is at. It then moves to the closest location of interest and again find the next location of interest that is closest to its new location disregarding all the locations of interest that have been visited already. The algorithm takes into account that the locations of interest do not form a complete graph, that is, there is a direct connection between every location. This is because locations of interest may be in different map segments and to get to it an overlap will have to be used. Therefore the shortest path is found using the Dijkstra's algorithm. The Dijkstra's algorithm calculates the distance of the shortest path and the path used to get to from one location to another.

The route planning is done deterministically, meaning that once the route is planned it is not change or updated. When the route planner finds a route to all the locations and passes it to the movement module it is not used again until the next tour.

4.3.1.2 Movement

The movement module receives a list containing the route that the robot needs to take in order to reach all the locations of interest and a list of all the locations of interest. The module uses the list of locations of interest to identify when the robot should stop and notify the application layer.

Upon receiving the lists from the route planner module the movement module requests for the map of the first location to be loaded by the map loader module. The map path is retrieved from the JSON that stores the map metadata (structure shown in Figure 3.4). From the same JSON the coordinates and the orientation are retrieved for the first location and the localization process is initiated.

After the robot is localized it commanded to move to the first location in the route list. When the robot reaches the location, it checks whether it is present within the list of locations of interest. If the location is not one that is to be visited, that is, it is not in the locations of interest list, the robot is commanded to move on to the next point in the route list, otherwise if it is a location to be visited, it alerts the application layer by sending the location that it has reached to the application layer.

The movement module handles overlaps differently from normal locations. If the location that it is at is an overlap, A, it uses the next map attribute of the overlap to find the map to be loaded. It then loads this map and localizes at the overlap, B, which has a key identical to that of A. All these attributes are found in the JSON storage structure hence XML files are no longer used by this module.

4.3.1.3 Map Loader

The map loader receives a link to a map and loads it using the “map_server” package in ROS as discussed in Section 4.1.3.

4.3.2 Application Layer

This section provides a description of the implementation on the tour guide robot application layer. Each of the modules in this layer are implemented as separate ROS nodes.

4.3.2.1 Tour guide control

The tour guide control module receives a list containing locations of interest from the interface relay module and sends it to the navigation layer, which will handle movement to each of the locations. The tour control by means of a ROS service waits to receive notifications that a location has been reached from the navigation layer. Once the notification is received, a path to the sound of the location is passed to the sound module. The tour control receives the duration of the sound and waits for its duration before sending the navigation layer the command to continue moving. The tour control also commands the interface relay to notify the user interface that it has reached a location. If navigation layer sends an end tour notification then the tour control ends the tour by relaying the end tour command to the user interface via the interface relay module.

4.3.2.2 Interface Relay Module

The interface relay module is implemented to communicate with the user interface. Since the user interface does not use the ROS framework the communication between the interface relay module and the user interface is established via a socket connection.

The interface relay module is implemented in two groups of functions, one group to receive messages from the user interface and send messages to the user interface via socket

connections and another group to receive from the tour control module and send to the control module using ROS services.

The function that receives from the user interface and the function that sends to the tour control module relay the list of locations of interest, while the function that receives from the tour control and send to the user interface relay tour progress notifications.

4.3.2.3 Sound Module

The sound module is implemented to publish a ROS topic that offers a service that receives a link to a sound. This module plays the sound using a system call and send the tour control the duration of the sound. This duration will be used by the tour control to wait at the position while the sound plays.

4.3.2.4 User Interface Module

The user interface was implemented on the Android platform. This was primarily because the user interface for the tour guide robot was a tablet running the Android OS. The implementation of the user interface used the model view controller (MVC) design pattern in order to abstract the application's functionality. By doing this, components, especially the controllers and models, could be reused assuming the view designs were to be changed.

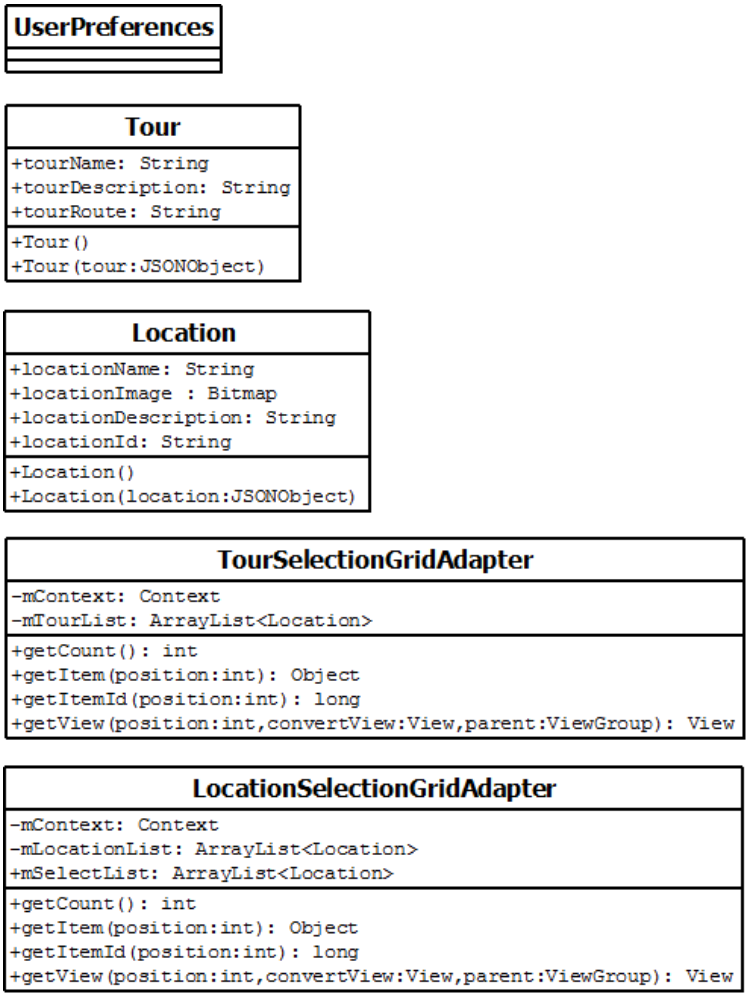

Figure 4.5 Class diagram for the models of the user interface

Figure 4.6 Class diagram for the controllers of the user interface

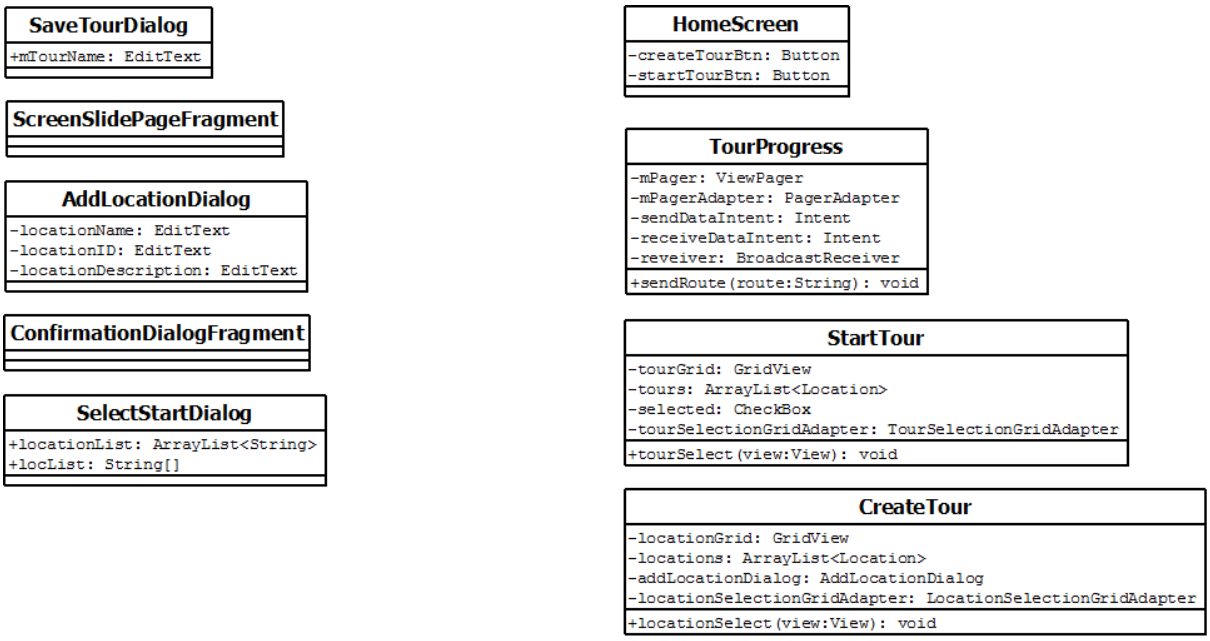

Figure 4.7 Class diagram for the views of the user interface

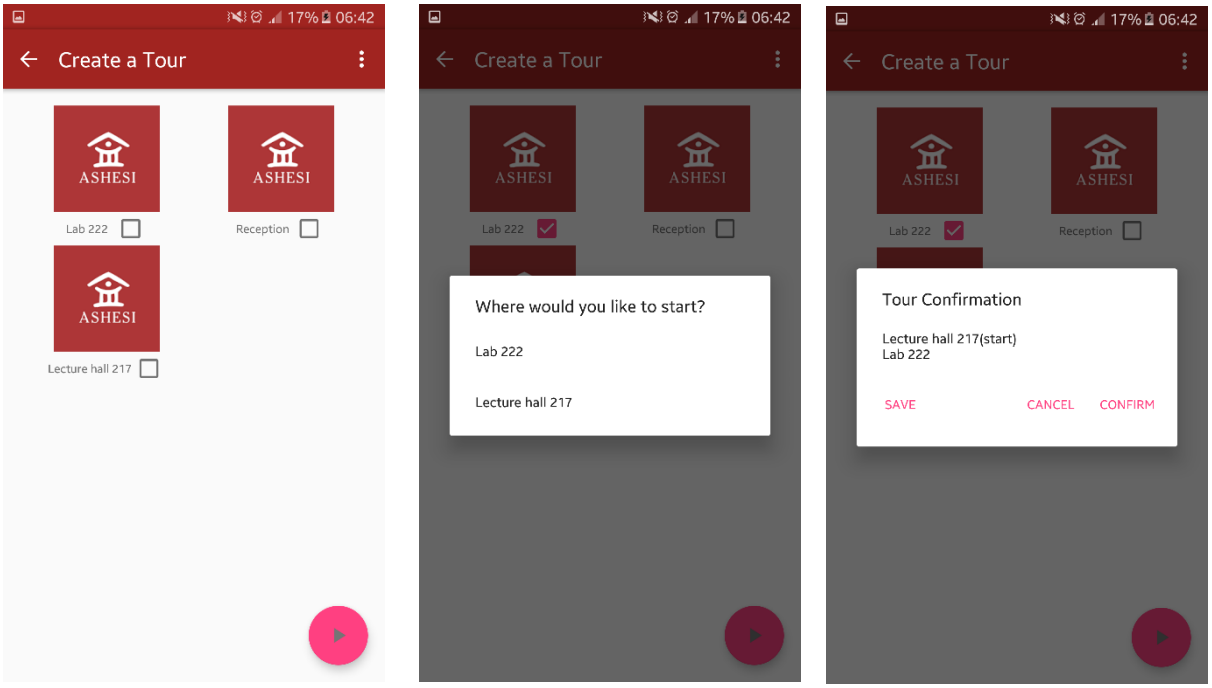

Figure 4.8 Graphical views of the user interface

Chapter 5: Testing and Results

5.1 Approach

During the development of the tour guide robot, tests were conducted in order to validate the functionality of the project. This section presents the tests cases, the test results and an analysis of the results.

5.2 Development Testing

This section outlines the tests that were carried out by the developer during the implementation of the project.

5.2.1 Unit Testing

Unit testing was conducted to check if individual functions in each of the modules perform the correct task and return the correct information.

Testing communication between the user interface and a server. This unit test was conducted to verify that the user interface could communicate with a server via a socket connection. In this test, a dummy server was used to receive information and would print the information on the screen. The dummy server was also able to send information back over the socket connection to the user interface, which would log the information. This test yielded the desired result thereby verifying that communication between the user interface and a server was possible.

Testing service in user interface. This unit test was conducted to verify that the service that would send information to and receive information from the workstation server in the background was working. The test was conducted by starting the service and periodically broadcasting information that was received and then printed to the screen. This test yielded the desired result thereby verifying that the service was working.

Testing storage of information to a file. This test was conducted to verify that user interface could store information and retrieve it in the correct format. The test was conducted by saving the string representation of a JSON to a file and retrieving. Upon retrieval the string was then converted into a JSON Object to verify that the string format was still correct. This test yielded the desired results thereby verifying the functionality of this function.

Testing item (location/tour) selection. This test was conducted to verify that an item selected on the screen was correctly being recorded. The test was conducted by displaying the information of the item that is selected on the screen. This test yielded the desired results thereby verifying the item selection functionality.

Testing temporary storage of selected locations. This test was conducted to verify that locations selected are stored as needed by the user interface. This test was conducted by logging all the selected locations. This test yielded the desired results thereby verifying the functionality of the user interface to temporarily save selected location.

Testing route planning. This test was conducted to verify the route planning functionality. This test was conducted by manually providing the route planning function with a list containing locations of interest. The route planning function was run and the output displayed and compared with the expected route. This test yielded the desired results thereby verifying the functionality of the route planning function.

5.2.2 Component Testing

This section describes the tests performed on components of the project to ensure they worked correctly.

Testing the Turtlebot movement. This test was conducted in order to verify that all the necessary parts worked together to move the Turtlebot. The movement involves map

loading, localization and moving from one point to another. This test was conducted by providing the movement component with a map, a start location and an end location. The results of each function was printed out to verify output. This test yielded the desired result for all the maps and locations used.

Testing the navigation layer. This test was conducted in order to verify the functionality of the navigation layer, which involved the retrieval of a set of locations of interest, the building of a route and the traversal of the given route. In order to carry out this test a list of locations of interest were manually passed to the navigation layer. The results of this test varied based on the locations of interest that were provided to the navigation layer. The problem identified was a flaw in the construction of the maps. Some maps were poorly built therefore creating false obstacles, which the robot took into consideration while planning a path. The existence of these false obstacles blocked the path to some location thereby hindering the movement process.

Testing start tour from user interface. This test was conducted to verify that a user could start a tour from the user interface. This involved the user selecting either locations to visit or a selecting a tour, starting the tour and the user interface sending the tour to the interface relay. This test was conducted by connecting the user interface to a dummy server and starting a tour using the interface. The test yielded the desired results thereby verifying that a tour could be started from the user interface.

5.2.3 System Testing

This type of testing was conducted on a fully integrated tour guide robot. This was done in preparation for user testing. In conducting this test the application layer and the navigation layer were linked, allowing communication between the two. In this testing a tour was started from the user interface. The test yielded varying results due to the

movement module in the navigation layer, which was unable to plan a path in a map of poor quality.

Chapter 6: Conclusion and Recommendation

This paper describes the implementation of a tour guide robot for Ashesi University College. However, in order to build this tour guide robot there needed to be an underlying navigation framework that would allow the robot to move around the university campus. Hence, the project involved the implementation of a generic navigation framework that would allow movement around the university campus and a tour guide application that used the framework.

This project has yielded a partially functional navigation framework. This is due to the poor quality of some maps created during the mapping stage of the project, which disables the robot from travelling to all locations in poor quality maps. However, in maps of better quality the robot is effectively able to move through them. The poor quality of some of the maps is caused by the instability of sensor readings from the Microsoft Kinect while mapping. The route planner of the navigation framework is fully functional and constructs the shortest route through a set of given locations of interest. The map loader is also fully functional.

The tour guide application meets the following requirements of the project:

- Able to create a new location with the user interface.
- Able to pass the locations of interests from the user interface to the tour guide robot.
- Able to pass data to the navigation layer.
- Able to pass a sound when a location is reached.
- Able to select and start a tour.
- Able to create and start a tour.

- Able to save a tour.

This project can be improved to make it a more effective system to be deployed on the university campus. Work that can be done in the future are;

- Improvement of the user interface. The user interface if a very important part of a tour guide. The current project allows the user to accomplish basic functionality, but is not very intuitive and interactive. Hence, a more animated approach can be taken in the development of a user interface, which will effectively communicate with users.

- Extended mapping of the university. In order to extend the scope of the navigation framework, more of the university's campus needs to be mapped.

- Additional applications. Using the navigation framework, other applications that depend on movement between locations can be built. Such as applications include delivery robots, messenger robots, among others.

In conclusion, this project has great potential to accomplish its goal of substituting the humans in the tour guide system on the Ashesi University College campus and can be even better with further development.

References

Burgard, W., Cremers, A. B., Fox, D., Hahnel, D., Lakemeyer, G., Schulz, D., . . . Thrun, S. (1998). *The InteractiveMuseum Tour-Guide Robot.* American Association for Artificial Intelligence.

Cormen, T. H., Leiserson, C. E., Rivest, R. L., & Stein, C. (2009). *Introduction to Algorithms.* Cambridge: The MIT Press.

Dellaert, F., Fox, D., Burgard, W., & Thrun, S. (1999). Monte Carlo Localization for Mobile Robots. *International Conference on Robotics & Automation* (pp. 1322-1328). Detroit, Michigan: IEEE. doi:10.1109/ROBOT.1999.772544

Faber, F., Bennewitz, M., Eppner, C., Gorog, A., Gonsior, C., Joho, D., . . . Behnke, S. (2009, September). The Humanoid Museum Tour Guide Robotinho. *The 18th IEEE International Symposium Robot and Human Interactive Communication* (pp. 891-896). Toyama: IEEE. doi:10.1109/ROMAN.2009.5326326

Mataric, M. J. (2007). *The Robotics Primer.* Cambridge: The MIT Press.

NilsBerg. (2015, October 12). *amcl*. Retrieved from ROS.org: http://wiki.ros.org/amcl

O'Kane, J. M. (2013). *A Gentle Introduction to ROS.* Independently published.

ROS. (2015, August 6). *Robot Setup*. Retrieved from ROS.org: http://wiki.ros.org/navigation/Tutorials/RobotSetup

Thomas, D. (2014, May 22). *Introduction*. Retrieved from ROS.org: http://wiki.ros.org/ROS/Introduction

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, pp. 1999-2005. Detroit, Michigan: IEEE. doi:10.1109/ROBOT.1999.770401

Appendix

A. Requirement gathering

A.1 Interview Questions

Below is a list of the questions that were asked in the interviews to gather the requirements for this project.

Researchers

1. Based on the goals, what has been accomplished in the project?
2. What are the features that have not been implemented?
3. What are the limitations of the tools being used?

Users

1. What do you think is important to see in Ashesi University College
2. What would you like to know about Ashesi University?
3. What do you think about taking a tour given by a robot?
4. What are some of the things that you think are interesting to see and know?
5. What is the best way to display information in such a project to a visitor?
6. What sort of interaction would a user expect when taking a tour?
7. Where do you take the visitors on a tour?
8. What do you tell the visitors on the tour?
9. What do you think about a tour guide robot?

A.2 Interview Observations & Insights

Researchers

- The major goal of this project was to have a robot tour guide for the Ashesi University College campus.

- In order to increase efficiency in terms of computer memory, the campus was broken down into smaller segments when mapping.
- Due to the involvement of smaller map segments, overlap regions were introduced, which would be used to join the map segments at runtime when moving from one map to another.
- Route planning needed to be done in order for the robot to choose which paths to take when visiting the locations chosen by the visitor in an effective manner.
- The map segmentation was carried out in order to increase mapping accuracy. While mapping large areas, the Turtlebot's odometry module had the tendency to lapse, hence causing it pose coordinates to be erroneous. This would affect the robot when it was sent to a location, because it would appear at the correct location in the virtual map, but at a different location in the real world. To reduce the occurrences of lapses while mapping, smaller section were mapped and then linked dynamically at runtime.
- Map details and information was stored in XML formatted files, which needed to be parsed at runtime in order to be passed to the modules that handled movement and map loading.
- An applet was used to generate the XML file, containing map details, which would be used to direct the robot. That is the XML file defined the route for the robot to take.
- The robot's movement was such that it could only move linearly through maps. This was because map segments had only two overlaps demarked for entry and exit. If a map was not linked to the entry point of another map segment it could not enter it, which would be the case for each map segment that was linked to the exit overlap of a map.

User

- One of the most crucial observations that arose in all interactions with the students and tour guides was their curiosity about whether the robot would “talk”. Recognising this, it shows the concern for interaction between the guide and the individual(s). Visitors did not merely want to be led around and shown exhibits, they wanted more; they wanted to interact.
- Speaking with these users shed light on the tour process, which did not only involve the movement from one point to another and the playing of a recording for a place, but also the sharing of experiences which were relevant at various points in time.

The insights gained from these stakeholder are used to formulate the use cases for the tour guide robot, as well as to determine functional requirements for in the project