Software Design Document

for

Archaeology Robot
Group 13

Yufeng Bai 1600095	
Dawei Geng 1219181	
Jun Chen 1206265	
Quang Khoi Nguyen 1187070	
Shikai Li 1214223	
Yunyao Yao 1203525	
Yatong Zhou 1204471	

Version 1.31

September 14, 2012

Contents

Revision History

Name	Date	Reason For Changes	Version
Dawei Geng	03 Sep 2012	Template	0.1
Yufeng Bai	06 Sep 2012	Chapter 7	0.2
Yufeng Bai	06 Sep 2012	Chapter 8	0.3
Yufeng Bai	06 Sep 2012	Chapter 9	0.4
Yufeng Bai	07 Sep 2012	Fix the layout	0.5
Jun Chen	08 Sep 2012	Chapter 1	0.6
Jun Chen	08 Sep 2012	Chapter 2	0.7
Yufeng Bai	09 Sep 2012	Fix the error	0.8
Khoi Nguyen	10 Sep 2012	Chapter 4,5	0.9
Jun Chen	11 Sep 2012	Chapter 3.1,3.2	1.0
Yufeng Bai	12 Sep 2012	Fix grammar and spelling errors	1.1
Yatong Zhou	12 Sep 2012	Chapter 6 and error fixing	1.2
Khoi Nguyen	14 Sep 2012	Chapter 4 review	1.31

Chapter 1

Introduction

1.1 Purpose and Scope

1.1.1 Purpose

The purpose of making this Software Design Document (SDD) is to give the details of the design of the archaeology robot and its system, which is designed by our group (group 13). In this document, we will give an overall description of architectural design, system organization and critical modules of the system. This document describes some general ideas of how do we design the system and how do we implement requirements into the system. This document will be given to the team members so that they can construct the system based on the requirements. Also this document will be used as a reference for further developments.

1.1.2 Scope

This Software Design Document (SDD) is to give details of overall design of archaeology robot. It will contain nine parts : Introduction which will give an overall description of the design, System Overview which will show the design of the system, System Architecture and Components Design, Architectural Description which also includes alternatives and rationale, Data Design, Human Interface Design, Resource Estimates and Definitions, Acronyms and Abbreviations.

1.2 References

This Software Design Document (SDD) references these files below:

1. Software Requirements Document (SRS) : this document demonstrate all the requirements for this project.

2. Software Project Management Plan (SPMP) : this document shows the details of how do we manage our project.

3. UML files : the class diagram show how do we design and implement our system.

1.3 Overview

This Software Design Document will be organised in a way that describe System, Architectural Design, GUI Design and Resource Estimates logically. In the beginning of the document, it will give an introduction of this document (Purpose and Scope) and System Overview. After that, the document will give some high level architectural design of the whole system, which include System Architecture and Components Design, Architectural Description. Then the document will move to the Data Design, which also includes Design Details. After the Data Design, it will also shows the design of Human Interface and Resource Estimates. Lastly, there are some Definitions, Acronyms and Abbreviations at the end of the document. This document will follow the developing stages.

1.4 Constraints

The follow restrictions, limitations, and constraints will affect the design and implementation of the system:

- The robot should be set up correctly before testing.
- The robot can not be damaged.
- The robot can not walk out of map.
- The robot can not push the wall and colliding is not permitted.
- The map should be set up correctly before testing
- The map is in fixed size (less than A1)
- There are some no-go zone on the map.
- The system must has save and load function.
- An acceptable latency is 300ms.
- There are some hidden wall in the map.

Chapter 2

System Overview

The main goal of the system is to let the robot can be used for survey an archaeological site containing the remnants of an ancient city on the premise that the robot can not be damaged.

Firstly, the system of the robot will use light sensor, touch sensor and ultrasonic sensor to locate some objects on the map, which include:

- walls
- hidden walls
- obstacles on the surface

Secondly, the robot which is used to survey an archaeological site is programmed by leJOS software. With this software we can design the functionalities of the robot correctly, so that it can detect every object on the map on the premise that the robot is not damaged. When the robot detect somethings, the system should deal with it by using a proper way which is described in the Software Requirements Document (SRS).

Thirdly, the operating software of the robot will have a nice graphic user interface (GUI) which contains:

- Control panel
- Map area
- robot information
- battery usage
- mode
- etc..

Lastly, the database of this project shall contain XML map files to store and update the survey map, the map shows :

- the whole map
- No-go zone
- detected walls
- detected hidden wall

- detected obstacles
- unexplored zone
- explored zone
- the boundary of the map
- initial point

In conclusion, the whole system of this project contains four parts:

- Client System which is installed in the robot for implementing its functionalities.
- User(Host) System which is used to control the robot and do things the clients want.
- GUI System is used to give a graphic user interface to the User System so that it is easier to be used.
- Database System which will store the survey result into a XML map.

With these four parts the system will work properly and do the task based on the requirements.

Chapter 3

System Architecture and Components Design

3.1 Architectural Description

The archaeology robot system we make will have four major parts that work together to achieve the goal. These four parts are Client System, Host System, Graphic User Interface System and Database System.

Client System: The Client System is installed in the archaeology robot. Basically the Client System will be implemented all basic functionalities of the robot. Also it is required to respond to commands which are sent from the Host System. Moreover the Client System should also include functionalities that can make the robot move according to received command.

Host System: The main purpose of the Host System is to control the movement of the robot so that it can survey the map autonomously and safely. To meet this goal, all commands that control the robot will be implemented into the Host System, which include movement, stop, slow down, etc. Also it will be able to update the database during mapping process. The Host System will also include function to change between manual mode and automatic mode.

GUI System: The purpose of the GUI System is the graphical interaction between the host and the robot. With GUI System, the user can keep track of the position of the robot. Also the GUI System will show all information about the map and robot and the user can send commands to the Client System via GUI System. Moreover the GUI System can save(load) files to(from) the Database System.

Database System The Database System is used to store the map of the survey area. The map is in XML format.

3.2 Component Decomposition Description

3.2.1 Client

The Client will use Blue-tooth to connect to the Host. It receives commands from the Host and convert them into the commands that the Client can use. After executing the commands it will return feedbacks to the Host for updating Database.

Blue-tooth Client The Blue-tooth Client allow the Host to communicate to the robot using a predefined interface. It encodes and transmit a command to the robot and waiting for a reply.

Robot Client The robot operators executes the command received from the host The Client's interface will define the method for which the Host side will call upon to implement desired motion and return desired values.

3.2.2 Host

The Host is the most important part in the system. It will be able to send commands to the Client (robot), also it can receive feedback from the Client (robot). Moreover, the Host shows every detail and change to the user by the GUI and it can update the database by sending commands. The components of Host include:

ManualController is the main part of the Host. It is used to control the robot to perform any actions safely and precisely. Also it need to deal with the feedbacks that are sent from the Client. So it need to analyse them and then update the Database, showing any changes on the GUI.

AutoController will be used when the robot is in auto mode. It will disable most of all commands from the users (except Mode change command). The AutoController will determine the safe performances by analysing the current map via ultrasonic sensor.

3.2.3 GUI

The GUI will contain two components which are:

Main GUI is the graphic interface for the host system which is used to show to the user. The main frame of it has many small components which act like functions. It includes buttons to control the robot, robot status, system status, save or load map and menu.

Map Panel is a component of the GUI that shows the surveying map. It is used to display the location of the robot and draw the real map into it, which means the map panel will display and draw everything the robot explored. After the survey, the map viewer will save the map into the database if it receives a 'save' command from the host.

3.2.4 Database

The Database component is used to store the map after survey (survey value). The map is in XML format.. It allows the user to save XML files in it, also it allows the user to load any XML files which is stored in the Database. The Database only receive commands from the Host.

XML files is used to store the 2D maps which are explored by the robot. Also some special zones will be stored into the map. The zones includes information of no-go zones, unexplored zone and explored zone. Zone will be represented as some blocks(pixels).

3.3 Detailed Components Design Description

3.3.1 Interfaces

Client

- Component Identifier: C-0

- Purpose(function): to detail method declaration and what needs to implement to control the movement of the robot such as moving, turning left, turning right, stop
- Subordinates: client.BluetoothClient and client.RobotClient.
- Dependencies: none
- Interfaces: none
- Data:The interface stores information related to the Client,

Controller

- Component Identifier: C-1
- Purpose(Function): to detail method declaration and what needs to implement for calls made by host.ui.UserInterface classes to the controller class
- Subordinates: host.Driver
- Dependencies: none
- Interfaces: none
- Data: none

Database

- Component Identifier: C-2
- Purpose (Function) : to detail method declaration and what needs to implement to create map, load map and save map
- Subordinates: database.XMLDatabase
- Dependencies: none
- Interfaces: none
- Data: none

Map

- Component Identifier: C-3
- Purpose(Function): to detail method declaration and what needs to implement to get and set the state of the map as well as get the position of the robot
- Subordinates: database.ArrayMap
- Dependencies: none
- Interfaces: none
- Data: none

User Interface

- Component Identifier: C-4
- Purpose(Function): to detail method declaration and what needs to implement to allow the user of changing the map and the database, tog notifying of found walls
- Subordinates: host.ui.GUI
- Dependencies: none
- Interfaces: none
- Data: none

3.3.2 Class Files

RobotController

- Component Identifier: C-5
- Purpose: initialising the system and coordinating the operations of the robot

  SRS requirement: touches almost all of the requirements
- Function:The class creates host.database.XMLDatabase, a host.MasterController, a client.Client and host.ui.GUI object.It creates client.BluetoothClient object to communicate to the client. It also saves and load a map into a XML file. It makes call to the AutoController for the next move.
- Subordinates: host.Controller, host.database.Database, host.ui.UserInterface and client.Client, host.AutoController, client.BluetoothClient, host.database.XMLDatabase, host.database.Map and host.ui.GUI
- Dependencies: host.Controller, host.database.XMLDatabase, host.ui.GUI, and client.BluetoothClient
- Interfaces: Controller, Database, User Interface, Client
- Data: storing the module objects and the flag variables to determine a simulation or real connection; object detection parameters and threshold; minimum move and rotation distances

ManualController

- Component Identifier: C-6
- Purpose: to control the robot manually and safely

  SRS requirement: R0001: Manual Control R0002: Safety under manual control
- Function: All of the basic movement control will be implemented in the class. The user issues a command from the GUI, which goes to the controller. The controller then compute what needs to do. After it figured out, it will send details instructions to the robot client.
- Subordinates: client.Client.
- Dependencies: host.Controller
- Interfaces: Controller
- Data: the position of the robot relative to the walls

AutoController

- Component Identifier: C-7
- Purpose: to autonomously exploring the map safely

  SRS requirement: R0004: Automatic Exploration, R0005: Avoiding danger
- Function: The method will take a Map file and then determine the robot?s current location, bearing and then possible moves. It always check if a move is safe before executing
- Subordinates: host.database.Map and client.Client.
- Dependencies: host.Controller
- Interfaces: Controller
- Data: the position of the robot relative to the walls

BluetoothClient

- Component Identifier: C-8
- Purpose: to act as a communication between the host and client.

  SRS requirement: R0001: Manual Robot Movement, R0004: Automatic Move Robot, R0005: Automatic robot exploration, R0011:Load and Save XML Files
- Function: When a command parameter is parsed into the method, the command is sent to client.BluetoothHost.
- Subordinates: client.BluetoothHost and host.RobotController
- Dependencies: client.Client.
- Interfaces: Controller, Client
- Data: none

BluetoothHost (RobotServer)

- Component Identifier: C-9
- Purpose: to act as a communication between the host and client.

  SRS requirement: R0001: Manual Robot Movement, R0004: Automatic Move Robot, R0005: Automatic robot exploration, R0011:Load and Save XML Files
- Function: opening a I/O stream to get input from client.BluetoothClient. Client.RobotServer remains in an infinite loop while waiting for a command.
- Subordinates: client.BluetoothClient
- Dependencies: none
- Interfaces: Client
- Data:none

Robot

- Component Identifier: C-10
- Purpose: to move the robot and to record the detected features

  SRS requirement: R0001: Manual Robot Movement, R0004: Automatic Move Robot, R0005: Automatic robot exploration, R0011:Load and Save XML Files
- Function: The robot client receives command from a client.BluetoothHost and then perform accodingly. If it detects any features of the area, it will return data to client.BluetoothHost.
- Subordinates: none
- Dependencies: client.RobotServer
- Interfaces: Client
- Data: the position of the robot, the speed and angles that the light sensor should rotate , the collected data

XMLDatabase (XML-ReaderWriter)

- Component Identifier: C-11
- Purpose: to be a map database for a host machine. Maps can be loaded and saved by the class.

  SRS requirement: R0011: Load and Save XML File
- Function: initializing a database.Map object and allows the creation of a map through the database.Map class. The class has method to get the state of the map such as pixel data and different zones. It is able to set and get the position of the client on the map. The current state of map can be saved to an XML file. A unfinished or finished map can also be loaded into XMLDatabase.
- Subordinates: host.database.MapStructure
- Dependencies: host.database.DataBase and host.database.Map
- Interfaces: Database and Controller
- Data: none

Zone

- Component Identifier: C-12
- Purpose: To store the state of each grid zone into 4 pixels (sub-grids)

SRS requirement: R0011: Load and Save XML File

- Function: initialising the 4 pixels as unexplored. The class can set and get the state of a pixel, which is represent as a pre-defined integer.
- Subordinates: host.database.Map
- Dependencies: host.database.Map
- Interfaces: Map
- Data: variables storing the states of the zone's four areas.

MapStructure

- Component Identifier: C-13
- Purpose: to create a map storing states of small grids in a two dimensional arrays

SRS requirement: R0011: Load and Save XML File

- Function:initialising the map to be the size of a grid, which is two by two. As the robot explores the map, it calls setBoarder method to increase the size of the map. The state of each pixel can be set and accessed through the setPixel and getPixel methods
- Subordinates: host.database.Zone.
- Dependencies: host.database.Map
- Interfaces: Map and Database
- Data: a two-dimensional array storing Zone objects

MapPanel

- Component Identifier: C-14
- Purpose: to display the graphical representation of the map

SRS requirement: R0007: Map Representation

- Function: The map is displayed using information from the database. The map shows the position of the robot and its traveled distance
- Subordinates: host.database.Map and host.database.Database.
- Dependencies: host.ui.UserInterface
- Interfaces: Map and Database
- Data: position of the robot and how far the robot has travelled.

GUI

- Component Identifier: C-15
- Purpose: an interface in the host machine to control and keep track of the robot

SRS requirement: R0008: Robot representation,R0009: Robot mode change, R0007: Map representation

- Function: manually controlling the robot, entering the autonomous exploring mode, updating the battery and signal status
- Subordinates: host.ui.MapPanel, host.ui.RobotController and host.ui.XMLDatabase.
- Dependencies: host.ui.UserInterface and host.Controller
- Interfaces: UserInterface and Controller
- Data: predefined representation of functionalities on the screen, the current state of the robot.

3.4 Architectural Alternatives

The other alternative design which is considered is letting the robot do the exploration without waiting command from the host. The idea is shelved because the robot's resource is not capable of performing the algorithm.

3.5 Design Rationale

The design of the system is layer architecture. The model is chosen for its support of separation and independence. The development is spread across teams with each team being responsible for a different part. The use of interface allows smooth integration without risk of incompatible method calls. If there is change in the interface, only the adjacent layer is affected so change is localised.

Chapter 4

Data Design

4.1 Database Description

The map is represented in XML format. It is created when the robot starts to explore.The map updates every time when the robot explores an area, detects a wall or a border of the survey area. The information which is stored in the array is translated and saved into XML. The XML file can be loaded as well. The detailed functionalities are as follow:

Creating a map The map is initialised to be a size of a grid, which is two by two. The size of the map is stored as digital number and the coordinates of the border are stored in an array list.

Recording the features of the area

- Clear Area: containing no features and is considered to be safe. The status and coordinate is saved if the area is explored.
- Wall Area: containing walls.The status and coordinate is saved if the area is explored.
- No-go-zone area: the robot is not permitted to enter the area. The status and coordinate is entered by the operator.

Recording the position of the robot The coordinates of current position are stored in an array list and are flagged as visited.

Saving The map which is explored so far can be saved.

Loading The previous map can be loaded,

4.2 Data Structures

The underlying data structure of the map database is a two dimensional array. Each element is the zone (grid)'s information, which is indexed by its coordinatting in the map. Each zone(grid) consisted of the information of its 4 sub-grids, which is stored in a two-by-two array

Chapter 5

Design Details

5.1 Class Diagrams

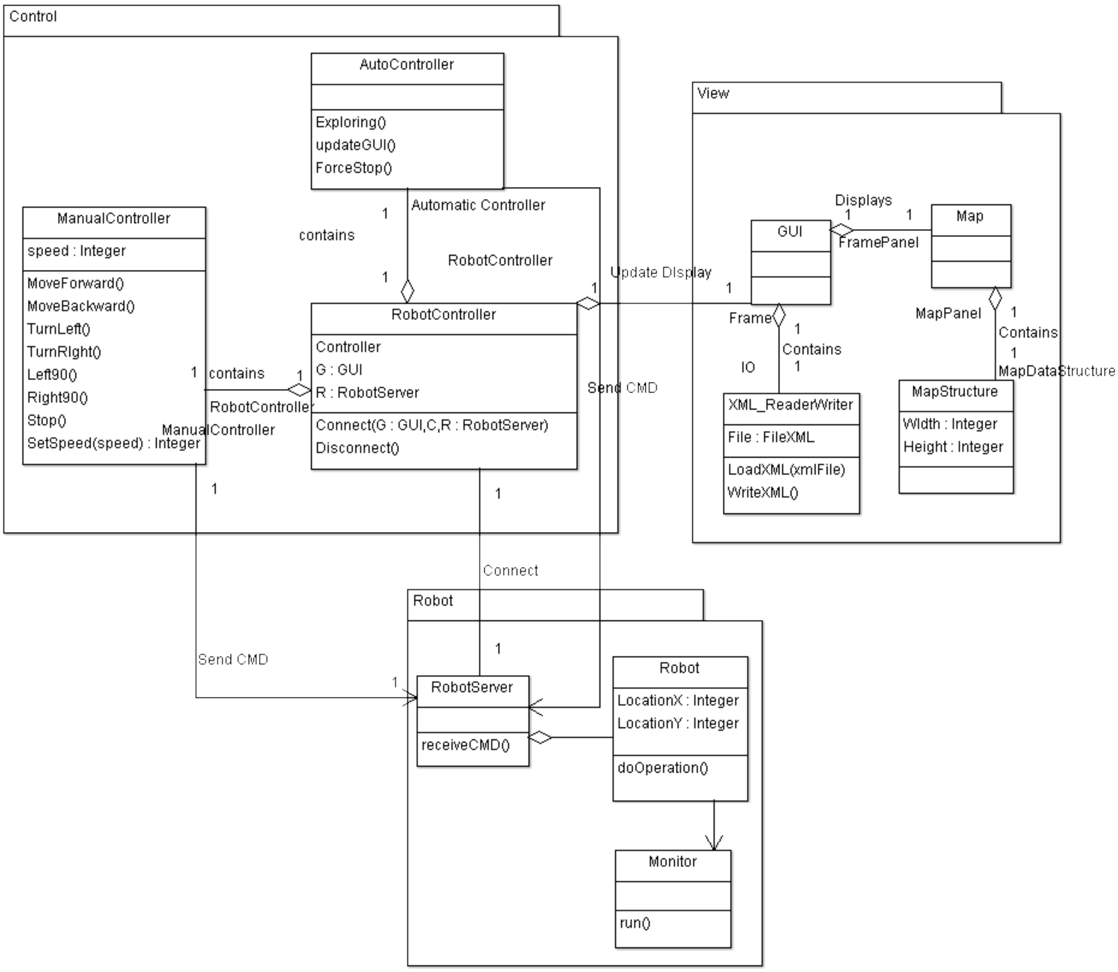

Figure 5.1: Class Diagram

The diagram above shows the package and class structures of the entire project. Note that the database is included the view section. The class model follows the classic MVC design pattern. Model is the controller part of the project, which controls the robot. GUI is the view part, displays the map in the run-time. Model is the robot itself mechanism, which is able to be accepted the command from the controller.

5.2 State Diagrams

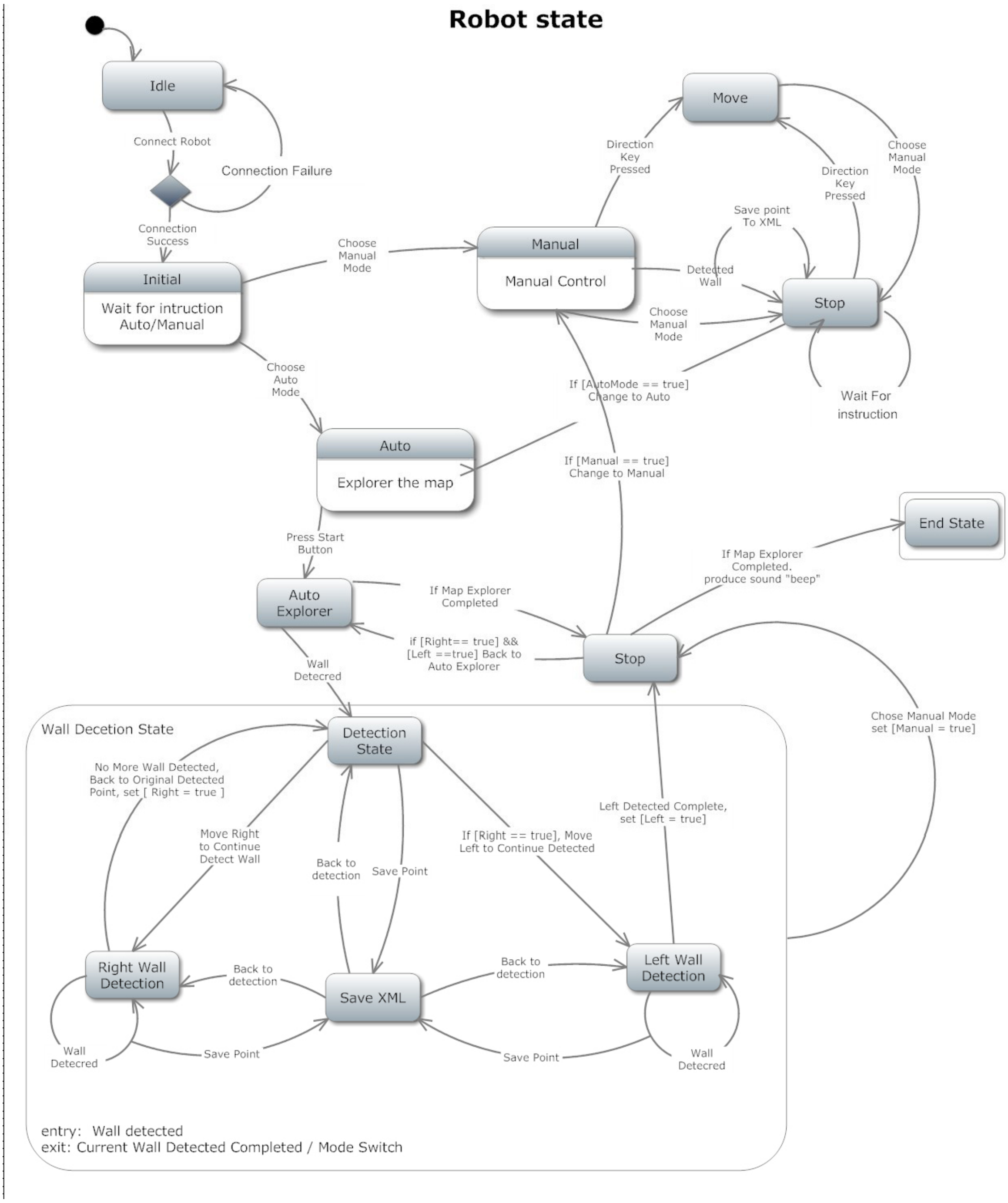

Figure 5.2: State Diagram

The state diagram above shows the all possible status of the robot. The robot started with initial state, after connecting, it jumps to "Idle" state, which means doing nothing but waiting for the commands from the host. After initial the map and establish connection. It wait operator to chose control mode. It transfers to either manual or auto control state to explore the unmapped areas. When transfer to auto mode(whatever in start or in the middle), robot will checks whether the map is fully explored. If yes, it reaches the end state. If not, the robot will do more exploration follow searching algorithms, if any wall is detected in the process, robot will record current location and start wall detection state, it will keep check the wall on left side until on more wall is found, and then check right side. After detected the wall, robot will come back to auto exploration mode. When transfer to Manual mode state, the robot can be control by operator, but for safety reason, if any wall is detected or go into no go zone in manual control, the robot will still pose its movement in until get next instruction that is safe to move. User can switch mode, and terminates exploring at any time.

5.3 Use Case Diagrams

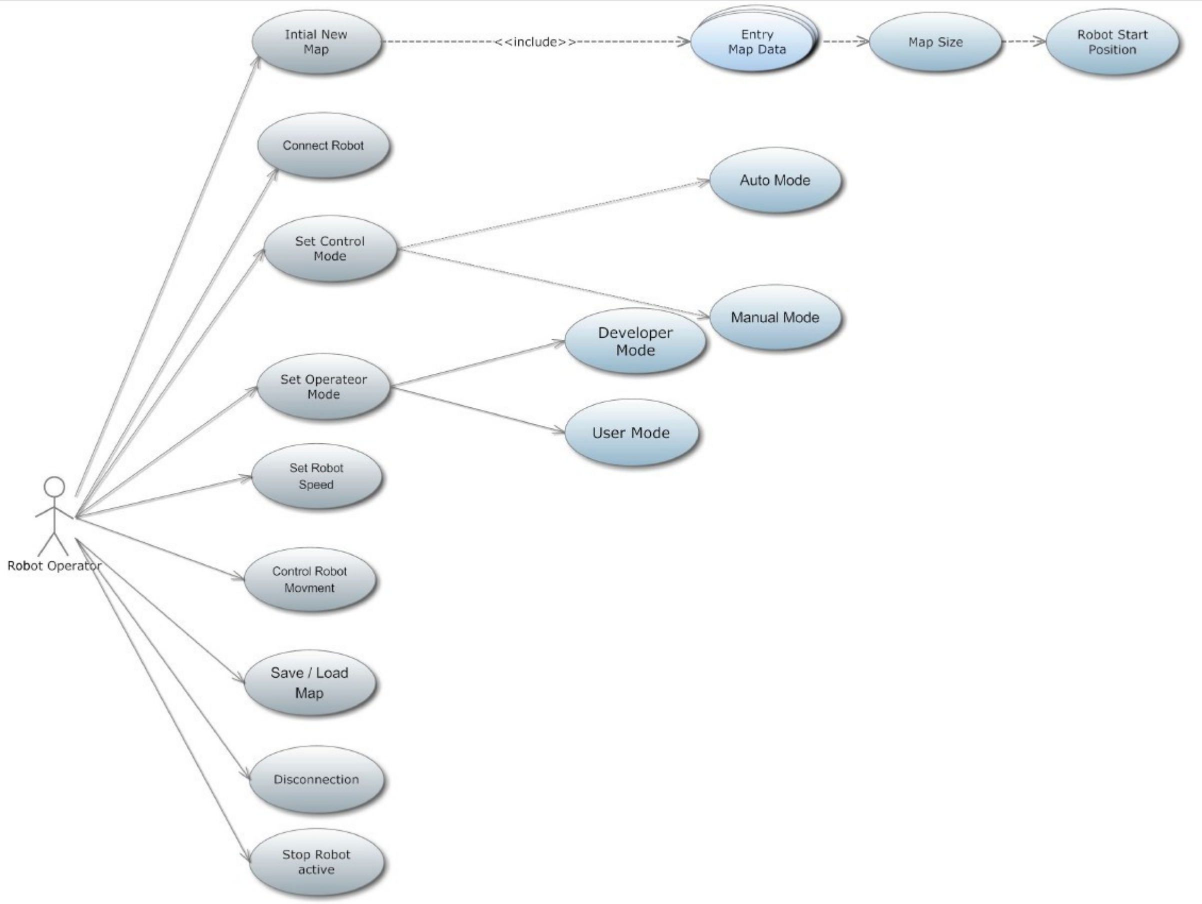

Figure 5.3: Use Case Diagram

The use case diagram above shows all the cases that user can operate on the graphic user interface. Start with initial map, this allow user to set initial value of map like map height, weight and start coordinate. Also a serial of instruction to to set up robot. That include establish robot connection, set operator mode (User/Developer), set control mode(Auto/Manual) and set robot movement speed. After robot set up, user can control robot through the control panel to explore the map. In addition user can save map or load map in the process. User also can chose disconnect with the robot or terminate this exploration at any time.

5.4 Interaction Diagrams

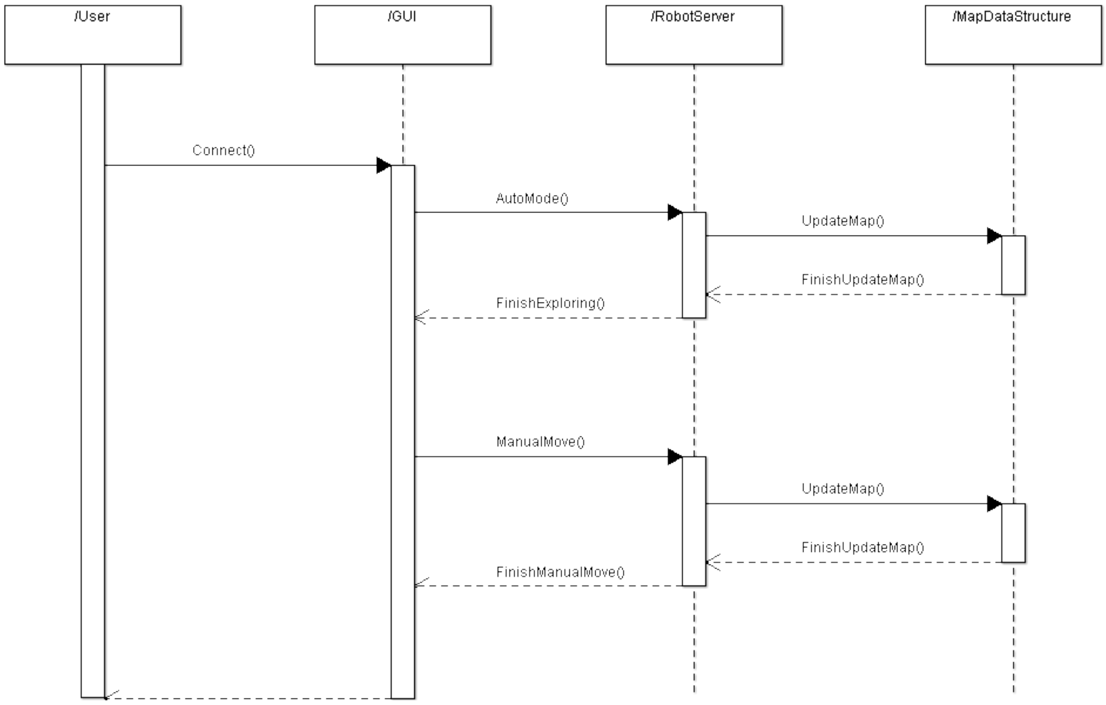

Figure 5.4: Interaction Diagram

The above diagram demonstrates how control is passed during the simple operation of the user manually moving the robot forward.
The GUI calls the hosts move function, which will do the required safeguard checks and then forward it onto the client, who will return the distance moved.
Afterwards the host will update the map data structure, so when the GUI updates the robot's new position and map display.

Chapter 6

Human Interface Design

6.1 Overview of the User Interface

The Graphical User Interface is used to communicate with the robot. The GUI is connected with robot, the client is able to control the robot by GUI. When the robot finishes its task, the client is allowed to switch off and disconnect with robot. In addition, the GUI demonstrates the status of the robot. The status includes the speed, the power and the location of the robot. The GUI also shows the obstacles, the "no-go" zone and hidden walls, which is able to be checked on the map window.

The GUI is able to demonstrate the follow functions for users:

- Save map to a XML file
- Load map from a XML file
- Change the control mode
- Demonstrate the Blue-tooth and Battery status
- Control the moving direction of the robot(i.e. forward, backward, rotating)
- Demonstrate the Icon of the map
- Demonstrate the Log Information
- Change the speed of the robot
- Demonstrate the coordinate of the robot

The GUI is displayed below:

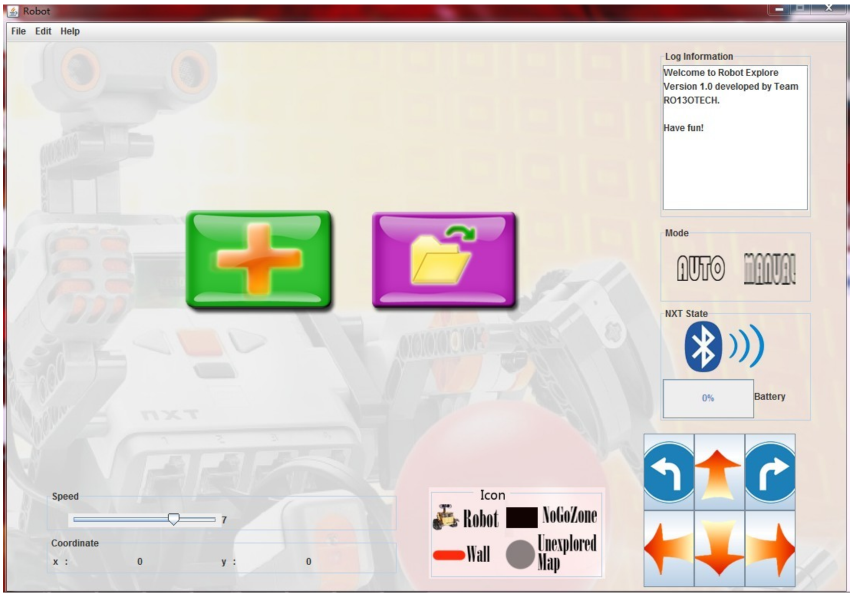

Figure 11: GUI

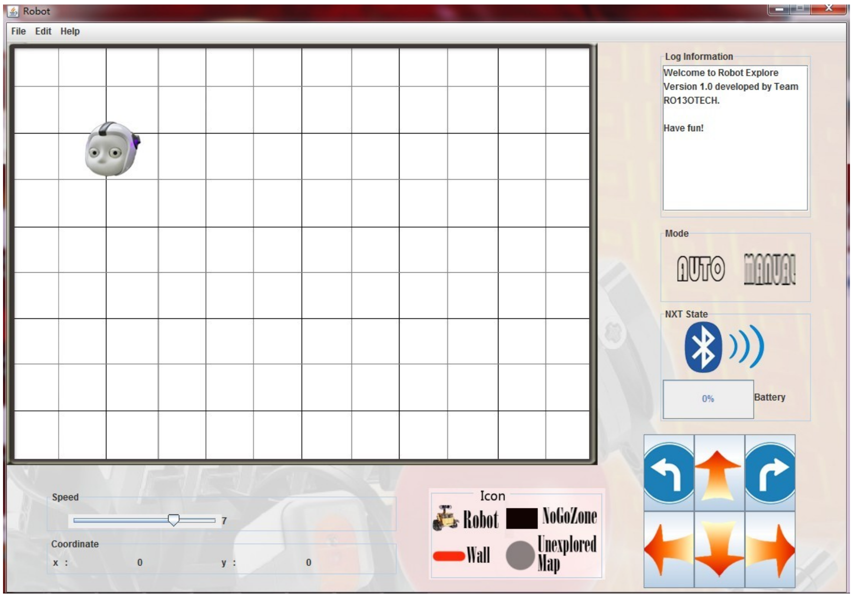

Figure 12: GUI WITH MAP

6.2 Detailed Design of the User Interface

6.2.1 Save map

The GUI includes a icon which is used to save the map from a XML file. When the client presses the File menu and choose the save option, there is a window jumping out, the client is able to choose the file where the client want to save.

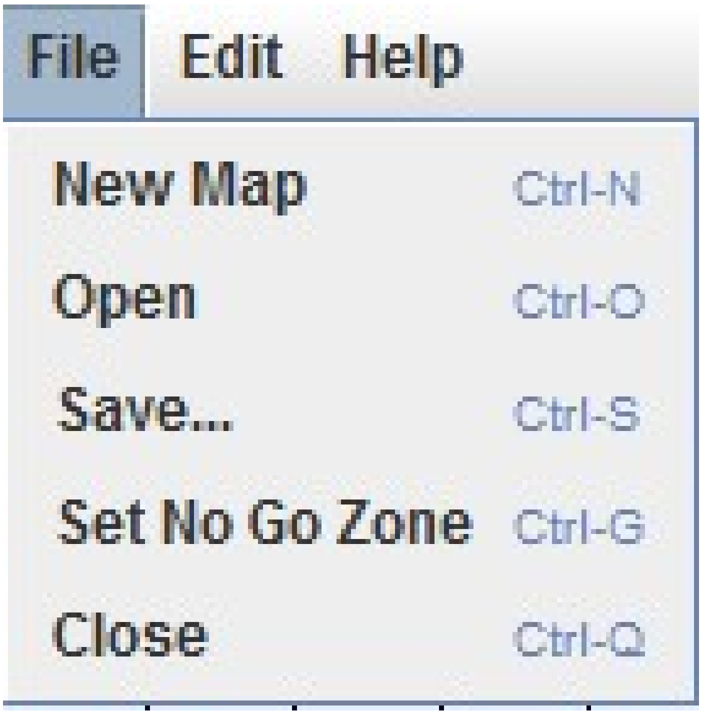

Figure 13: SAVE MAP

6.2.2 Load map

The GUI includes a icon which is used to load the map from a XML file. When the client presses that icon, there is a window jumping out, the client is able to choose the map from there.

Figure 14: LOAD MAP

6.2.3 Change the control mode

The client is allowed to change the control mode by pressing the mode button. When the client presses the Auto mode button, there is a widget jumping out. Then if the client presses ”yes”, the robot changes to the Auto mode. The client is allowed to do the same operation to change to the Manual mode. The Log information display which mode it is currently.

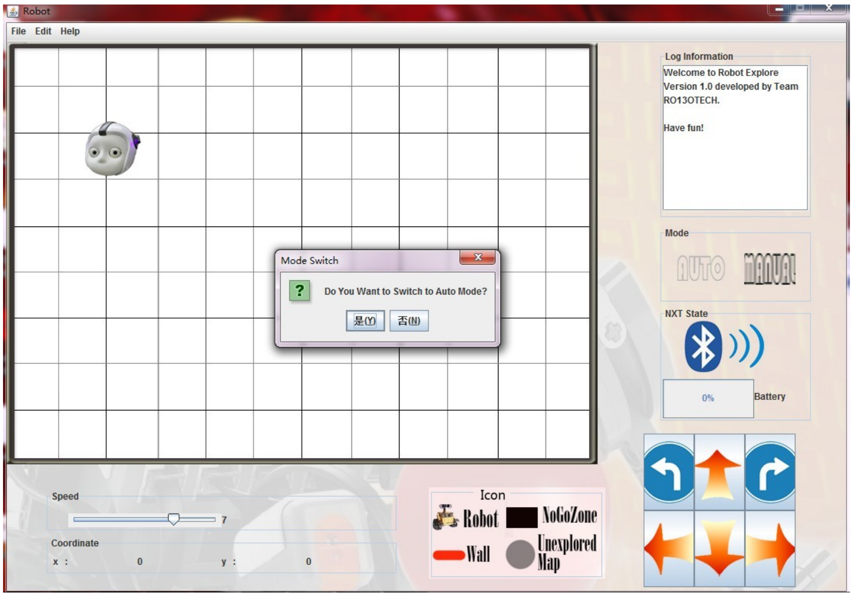

Figure 15: Change Mode

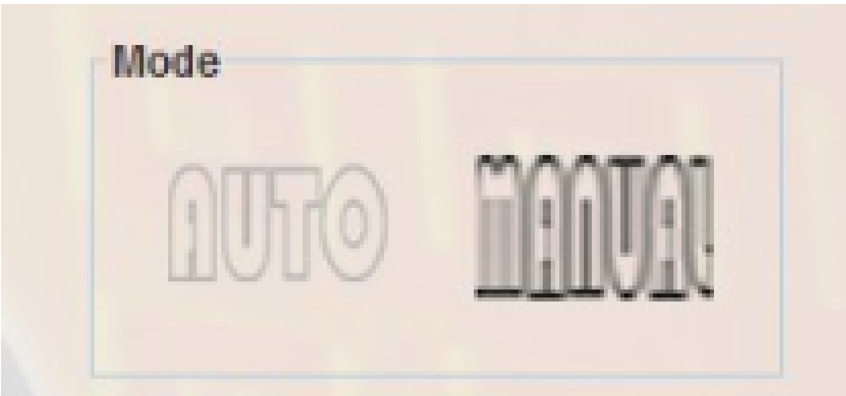

Figure 16: Change Mode icon

6.2.4 Demonstrate the Blue-tooth and battery status

The Blue-tooth and battery status are displayed on the GUI, if the Blue-tooth loses the connection, the log information board will demonstrate the lose information. The power of the battery is displayed by the percentage number and the colour of the battery bar will change along with the battery level.

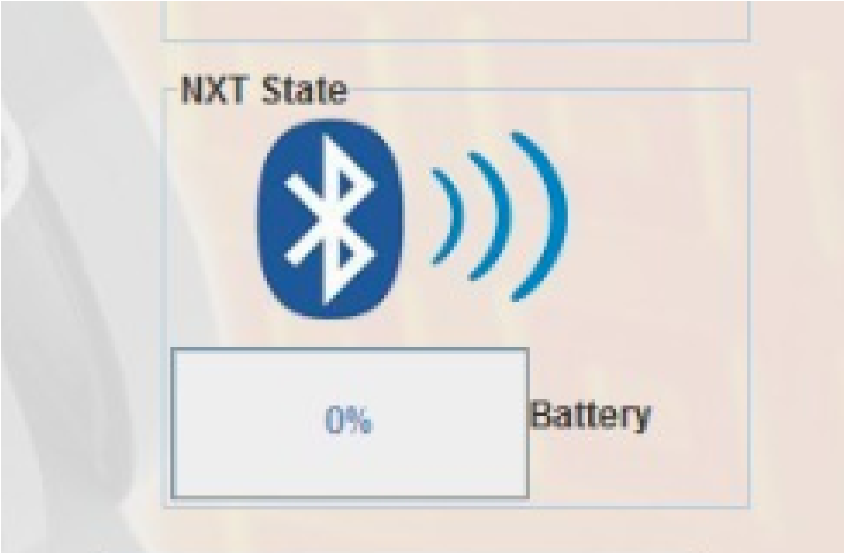

Figure 17: Blue-tooth and battery demonstration

6.2.5 Control the moving direction of the robot

The robot is able to be controlled by GUI when the robot is changed to the Manual mode. The client is allowed to press the arrows to move the robot. The robot is able to move forward, backward and rotate(include rotate 90 angles and rotate 360 angles). The moving direction controlling only can be used in Manual mode.

Figure 18: Moving control

6.2.6 Demonstrate the icon of the map

The robot and map information are required to be displayed on the GUI. It is necessary to use different icons to demonstrate the different map information.

Figure 19: Icon of the map

6.2.7 Demonstrate information of the GUI on the log information board

The Log Information board displays the information of the GUI. For example, the Log Information board demonstrates the version and the developers of the robot. If the client changes the control mode, the board displays which mode it is now. The board also is required to indicate the connection information.

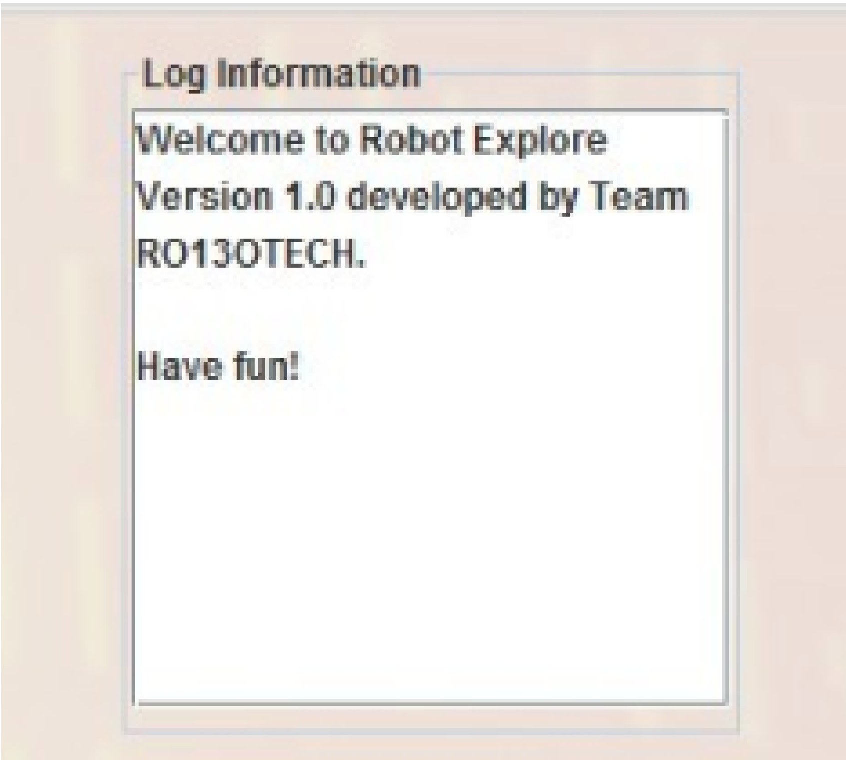

Figure 20: The logging information

6.2.8 Control the speed of the robot

The GUI includes a speed bar and one specific number to control the speed of the robot. The client is allowed to change the speed of the robot by change the number of the bar. The larger number means the faster the robot is. If the number is zero, it means the robot stops immediately.

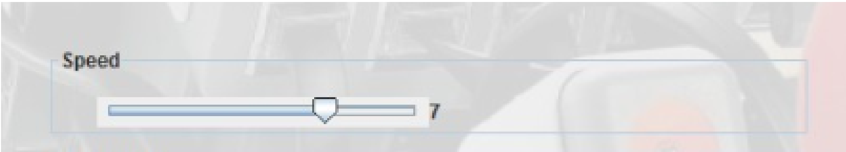

Figure 21: The speed controlling

6.2.9 The location of the robot

The GUI is required to display the specific location of the robot. The map is made up of a lot of grids. The location of the robot is able to display by coordinate, which is more accurate. When the robot moves, the coordinate will changes immediately.

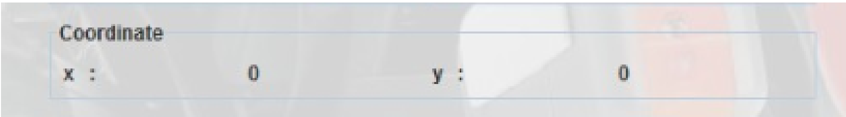

Figure 22: The coordinate of the robot

Chapter 7

Resource Estimates

The resource of the project is estimated according to the Project Description and the client's requirements in every week's meeting. All hardware and software which are used in the project are demonstrated below.

7.1 Hardware

7.1.1 Robot

Name: NXT Robot includes all components of the robot
Function: The robot is used to search the map and find all the obstacles, hidden walls and "no-go" zone. All operations are required to finish by controlling the robot.

7.1.2 Host

Name: PC
Function: The PC is required to demonstrate the searching information of the robot. PC is also used to control the robot on Manual Mode. The programs are uploaded to the robot from the PC.

7.1.3 Connection

Name: Blue-tooth and USB
Function: The Blue-tooth and USB are all used to connect with the robot. The USB is limited by the length of the cable. The Blue-tooth is limited by the uploading speed. In this project, the USB is used to upload the program and the Blue-tooth is used to implement the manual control.

7.2 Software

7.2.1 Operation Environment

Name: Mac, Linux and Windows
Function: This project is not providing the working environment. Any system is able to develop the programs for the robot.

7.2.2 Developing language

Name: Java, latex
Function: The codes are allowed to write in Java language, which is convenient to be read

by any developers. The documents are required to write by Latex, then generating the pdf documents.

7.2.3 Developing tool

Name: Eclipse
Function: Eclipse is a better tool to make the program for this project. The Eclipse is used to write the operation commands for the robot and draw the GUI for this project.

7.2.4 Robot Software

Name: leJOS 0.9.1
Function: The leJOS is used to control the robot and developing tool is required to support the leJOS 0.9.1.

7.2.5 Testing tool

Name: JUnit
Function: The JUnit is used to debug the code.

Chapter 8

Definitions, Acronyms, and Abbreviations

8.1 Acronyms and Abbreviation

Acronyms/Abbreviation	**Description**
API	Application Programmable Interface
BT	Blue-tooth
GUI	Graphical User Interface
PC	Personal Computer
SRS	Software Requirements Specifications
SPMP	Software Project Management Plan
UML	Unified Modelling Language
USB	Universal Serial Bus
XML	eXtensible Markup Language

Table 1: Acronyms/Abbreviations

8.2 Definitions

1. API: A set of classes and interfaces which are used to make the program for the robot include PC API and lejos API.

2. BT: A device to implement the wireless connection.

3. GUI: A interface which is generated on the PC is used to send the commands to the robot.

4. PC: A device which is used to make the program for the robot and build the GUI.

5. UML: A modelling language is used to build the diagrams for the project.

6. USB: A device to implement the wired connection.

7. XML: The data structure is used to store the map information.