User Interface for a Group of Mobile Robots

Liam Poli
20367119
School of Mechanical and Chemical Engineering
University of Western Australia

Supervisor: Professor Thomas Bräunl
School of Electrical, Electronic & Computer Engineering
University of Western Australia

Final Year Project Thesis
School of Mechanical and Chemical Engineering
University of Western Australia

Submitted: June 3rd, 2013

Project Summary

The field of robotics has progressed significantly over the past decade. New robust systems can now be used in challenging applications such as military reconnaissance, hazardous environment exploration and search and rescue. Current technology is based around the teleoperation approach which requires at least one human controller to monitor each robot. This approach however cannot be extended to multi-robot control. For a single operator to effectively control multiple robots, an increase in robot autonomy and an advanced multi-robot interface is required. State of the art robotic systems use multiple robots to complete complex and challenging missions that would not be possible with single robot systems. This project covers the design, construction and testing of a multi-robot user interface.

The system design was based on modern Real Time Strategy (RTS) video games. A large emphasis was placed on creating a user friendly system centred on a user experience based design. The system built provided a single user an effective means of controlling multiple robots; with high compatibility with ROS (Robot Operating System), the current standard robotic operating system used by the majority of research institutions. ROS compatibility will allow the product to be easily integrated into other robotic systems.

Evaluation was carried out by user testing and heuristic evaluation. The results show that for a single robot users had far superior control using the interface when compared to a simple teleoperation system. Effective multiple robot control was demonstrated by users for two robots through exploration of an unknown environment. The project results were used to develop a set of design guidelines to be used for future multi-robot interface design.

Acknowledgments

The author would like to thank his supervisor Professor Thomas Bräunl for his advice, guidance and for the opportunity to undertake in a very rewarding experience.

The author would like to thank all the users who donated their time to be involved in the user testing. Their contribution and suggestions that helped increase the quality of the final system.

Finally, the author would like to thank his friends and family for their support and encouragement during the course of this project.

Publications

Reid, R., Cann, A., Meiklejohn, C., Poli, L .& Braunl, T. *Cooperative Multi-Robot Navigation, Exploration, Mapping and Object Detection with ROS.*

To be presented at the 2013 IEEE Intelligent Vehicles Symposium, 26 June 2013.

Table of Contents

1. Introduction

The field of mobile robotics is both a promising and challenging one with potential to greatly advance many industries including mining, defence and consumer robotics. Current research is focused on developing effective complex multi-robot systems. With robots lacking advanced problem solving and reasoning skills, human supervision is still required to complete complex tasks. Thus a critical component of such a system is the user interface.

A robotic interface is the mechanism a human uses to interact with the robotic system, normally run on a computer and being called the master. Current multi-robot interfaces available are predominately designed for experts in robotics, making them near impossible to use for an average user. For a multi-robot interface to enter the commercial market, more user friendly systems are required.

The primary objective of this project is to research, design, construct and then evaluate a multi-robot user interface. The system created will focus on improving the state of the art by creating an interface which is:

1) User friendly, aiming to create a system that can be used by operators with limited robotics experience.
2) Compatible with current research robotic systems with the final goal to release the system to the robotic community.
3) A single user system.
4) Easily expandable so that other research intuitions can easily use the system to further the HRI field.
5) Modelled after popular Real Time Strategy Video Games.

1.1 MAGIC2010

The University of Western Australia's (UWA) robotic fleet consists of five robots, named WAMbots. Originally developed as an entry to the Multi Autonomous Ground-robotic International Challenge in 2010 (MAGIC 2010), it earned the team fourth place. The main objective of the competition was to develop a robotic system that could cooperate autonomously over extended missions. The challenge required two human operators to control three or more robots and to build a map and locate simulated bombs (red barrels) and hostile humans (Reid & Braunl, 2011).

The WAMbots were based on a Pioneer 3AT outdoor robot platform, with an automotive core-2 duo PC under Windows XP, three laser scanners (Sick LMS, Sick Ibeo and Hokuyo), a Qstarz GPS, an Xsens IMU, two digital cameras for teleoperation and an object identification/tracking system (Reid & Braunl, 2011). The system relied on sponsored military and commercial software, which at the completion of the project had to be returned and is now unavailable. The next generation system will implement the features of the original system using fully open source software. Future projects hope to further develop the foundation laid out by MAGIC2010.

1.2 ROS

ROS (Robot Operating System) is an operating system designed specifically for robots. Originally developed by Stanford University, the system is now supervised by Willow Garage. ROS provides hardware abstraction, device drivers, libraries, visualizers, message-passing, package management and more. It is released under the BSD licence and is open source, making it free for research use (Willow Garage, 2012).

1.2.1 Node, Topics and Services

ROS works by running a series of nodes or software modules. A full system can include nodes that enable mapping, navigation and communication. Nodes communicate with each other via topics, which a node can publish or subscribe too. ROS Messages are the data types that are sent along topics. Services can be used as nodal functions and node parameters can be set and changed in real time (Willow Garage, 2012).

When a node subscribes to a topic, the topic will begin sending data to that node. When a node publishes to a topic, every other node subscribed to the topic will now receive the incoming data. The advantage of such a system is that nodes can run independently of each other meaning a node can be stopped or changed without affecting the overall system. This is of importance when dealing with a robotic system that is made up of many sensors that can possibly fail. A distributed system such as ROS will allow the system as a whole to continue if one node crashes.

1.2.2 ROS Packages

ROS has a large community and provides many free open source packages that can be easily integrating into a robotic system. The standard package execution recommended by ROS is through the use of ROS launch files (Willow Garage, 2012).

1.3 Human Robot Interaction

Human Robotic Interaction (HRI) deals with the science of studying how humans, the user, interact with robotic systems. Current robotic technology is not yet advanced enough to operate without human intervention. Robots have their advantages; they don't grow tired, can perform boring and mundane tasks at high levels of consistency and can operate in hazardous environments. However, they lack advanced decision making and reasoning, skills at which humans excel at. Thus an optimal system will involve the cooperation of robots and humans working together. To facilitate efficient and optimal communication between the two, advanced robot interfaces are required.

Most current robotic systems rely on the teleoperation approach, in which a user controls the robot via a video feed and a controller. As humans can only process a finite amount of data at one time, effective control of multi robots using a pure teleopoeration approach is near impossible. The phenomenon of computational overloading explains how a user can only process a limited amount of information at one time; to focus on more than a single video feed will very quickly overload a user (Olson, et al., 2012). The limitations of control via a video feed are discussed in (Chen, et al., 2006). To implement a non-teleoperation-based system an increase in robot autonomy is required, providing a system that can be controlled with less user concentration.

The computational load on a user is higher when using teleporting compared with controlling an autonomous system (Dixon, et al., 2003) and (Schipani, 2003). However this is dependent on the reliability of the autonomous system; an unreliable system can possibly produce a higher cognitive load on a user compared with a teleoperation system (Dixon & Wickens, 2004).

1.3.1 Situational Awareness

Situation awareness (SA) can be defined as "the perception of environmental elements within a volume of time and space, the comprehension of their meaning, and the projection of their status in the near future" (Endsley, 1988). From this, five forms of situation awareness can be further defined: location awareness is the knowledge where the robot is located within an environment; activity awareness it the understanding of the progress the robot is making towards completing its mission; surroundings awareness pertains to obstacle avoidance, namely, the understanding of potential obstacles surrounding the robot; status awareness refers to the understanding of the status of the robot including; battery level, hardware status and the current mode of the robot; and overall mission awareness is defined as the understanding of the progress towards the overall mission objective (Gomez, 2010). An effective robot user interface must maximise all forms of SA.

1.3.2 Measuring Situational Awareness

Due its complex nature, measuring and quantifying a user's SA is difficult. Various methods have been proposed in the literature. Objective measures compare user perceptions of a feature to some constant real feature within an environment. Subjective measures ask users to rate their own SA, while performance and behavioural measures infer SA from task outcome (Endsley & Garland, 2000). The LASSO technique (Location Activity Surroundings Status Overall Awareness) involves getting users to "think aloud". This technique does not require users to rate themselves and thus is less subjective and is not based on task performance (Drury, et al., 2007).

1.3.3 Scalability

System scalability refers to an interfaces ability to still remain effective for an increasing number of robots. B. Trouvain and H.L. Wolf (2003) used a simulation of a multi- robot scenario invloving mapping and video feeds with a goal setting control structure. User testing showed eight robots were the optimal number; more than this resulted in a decrease in a user's situational awareness.

1.3.4 Human Computer Interaction

Human Computer Interaction (HCI) is a well-developed field of computer science that deals with the interactions of a human (user) and computers. HCI provides theories to help with interface development that explain cognitive aspects such as attention, memory, perception and recognition.

The human mind is limited to only focusing on one distinct point at a time; to monitor multiple points the mind will jump from one to the other. To aid in a user's attention management, information should be ordered and structured. This can be achieved using perceptual boundaries such as windows and tables, adding colour when an item needs to be drawn attention to and using sound and flashing lights to confirm user actions and to help divert attention. Keeping the visual design crisp, clean and free of clutter and the interface will help with user attention management (Sharp, et al., 2011).

Memory in the human mind involves two main processes: encoding followed by retrieval of knowledge. Studies have shown humans are much better at recognising things versus recalling things. Research has also shown the human mind is much better at remembering images rather than words (Sharp, et al., 2011). The design implications of such findings imply recognition should always be used over recall. An example of this would be when a user is required to selection an action. Where possible always show a list of possible actions should always be shown as opposed to having the user remember the name of the action.

Icons are used in interface design to condense information and help define a overall visual design. The use of hints, such as text overlays which appear when the user hovers over the icon can help a user comprehend icons they do not immediately recognise (Sharp, et al., 2011). User frustration will make a user inefficient and reduce their responsiveness to the interface. Common causes of user frustration include the interface crashing, the system not doing what the user expects, not providing sufficient information to enable the user to know what to do next, having vague or condemning error messages, requiring users to carry out too many steps to achieve a simple task and a noisy, gimmicky or patronizing visual appearance (Sharp, et al., 2011).

Usability in the context of HCI refers to how user friendly a program is or how easily a user can effectively use the system (Bevana, et al., 1991). The techniques of increasing a system's usability are centred on using continuous evaluation methods throughout project development. The standard evaluation method recommends using a combination of user testing and heuristic evaluation (Sharp, et al., 2011). User testing involves getting future users to trial the system and heuristic evaluation attempts to produce empirical data by ranking the interface against a standard set of evaluation criteria, such as Nielsen's heuristics (Nielsen & Molich, 1990).

1.3.5 Evaluation

System evaluation is a critical part of the software design process. HCI recommends evaluating a user interface with a combination of heuristics, metrics, user testing and expert involvement (Sharp, et al., 2011). Interface designers from ROS recommend using usability heuristics as outlined in Nielsen (1995). For example Steffi (2013) provides a set of usability heuristics specially tailored to robotic interface design. Design metrics can also be used to help guide development and conduct evaluation. Goodrich and Olsen, Jr (2003) provide principles for efficient human robot interaction. While Steinfeld, et al (2006) produced a comprehensive set of metrics defined in five categories: navigation, perception, management, manipulation and social. Design and evaluation of a multi-robot interface is discussed by Trouvain and Wolf (2003).

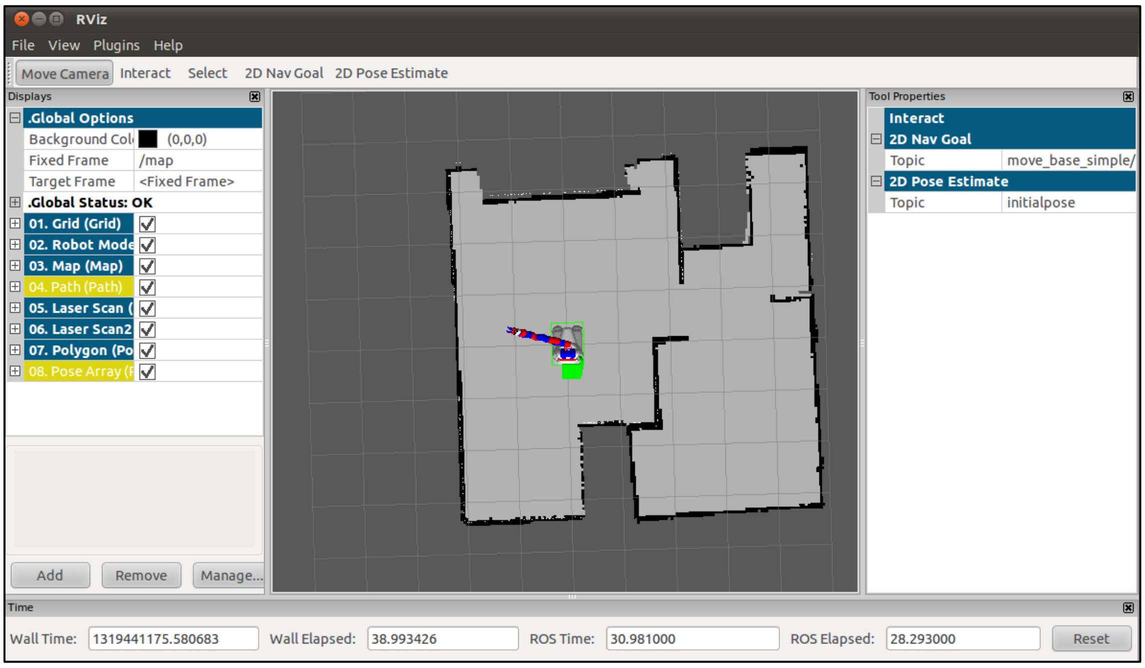
1.4 Real Time Strategy Games

This project will attempt to use a Real Time Strategy (RTS) video game paradigm to aid in the design process. RTS games are a genre of video games in which a player commands units and structures to compete objectives in real time. The fit to a robot interface is apparent: both an RTS game and a multi-robot system require users to control multiple units to complete a mission. The genre has been around for almost two decades, which has given it time to mature and develop a highly refined control style. Researchers at Stanford University developed a robot interface based on the RTS interface paradigm (Jones & Snyder, 2001). The system was developed as a proof of concept and still requires considerable work to become an effective control system.

Key RTS game features were determined by analysing popular games. Units are moved by first selecting the unit followed by giving the unit a goal, called a waypoint. When a unit is selected, the units relevant status and other important information is displayed in a control panel. The camera is positioned in a bird's eye view of the world with features allowing easy navigation. Data is displayed in a highly efficient manner with extensive user of colour, icons and structured visual layouts.

1.5 Current Systems

Rviz is a 3D visualization tool developed for ROS and is capable of displaying maps, robot positions, cost maps and other common ROS topics (Spaepcke, 2013). Rviz is capable of displaying the output from multiple robots; this makes it an effective data visualisation tool. However the control system has low usability and as it lacks adequate control mechanisms and robot feedback, control of multiple robots is not possible. Figure 1.1 shows a screenshot of Rviz.



.Global Options	
Background Col	■ (0,0,0)
Fixed Frame	/map
Target Frame	<Fixed Frame>
.Global Status: OK	
01. Grid (Grid)	✓
02. Robot Mode	✓
03. Map (Map)	✓
04. Path (Path)	✓
05. Laser Scan (	✓
06. Laser Scan2	✓
07. Polygon (Po	✓
08. Pose Array (F	✓

Interact	
2D Nav Goal	
Topic	move_base_simple/
2D Pose Estimate	
Topic	initialpose

Wall Time: 1319441175.580683 Wall Elapsed: 38.993426 ROS Time: 30.981000 ROS Elapsed: 28.293000

Figure 1.1 A screenshot from Rviz showing the localised position of a single robot within a map (Fraunhofer IPA, 2011).

The literature provides a few notable examples of state of the art multi-robotic user interfaces and four examples of such systems were analysed. All these interfaces can be classified as map-centric systems meaning the central control structure is implemented using map and not teleoperation through a video feed.

Figure 1.2 shows a system developed by the University of Pittsburgh and Carnegie Mellon University (Wang, et al., 2008). Developed and tested on simulated robot data the system is capable of controlling multiple robots. Control is facilitated through the setting of waypoints on a map with a video camera feed present for each robot. No robot status feedback is given and the visual design poorly structured. As the system was only tested on simulated data, no conclusions can be drawn on whether it will work in a real world scenario.

A state of the art multi-user, multi-robot interface was developed in 2010 by the University of Rome La Sapienza (Gomez, 2010), shown in Figure 1.3.The interface built was capable of controlling four robots and was developed for search and rescue applications. Although it is working and has been tested, the system is limited to control of four robots, lacks adequate feedback mechanisms and refined visual design, and has deficiencies in usability features.

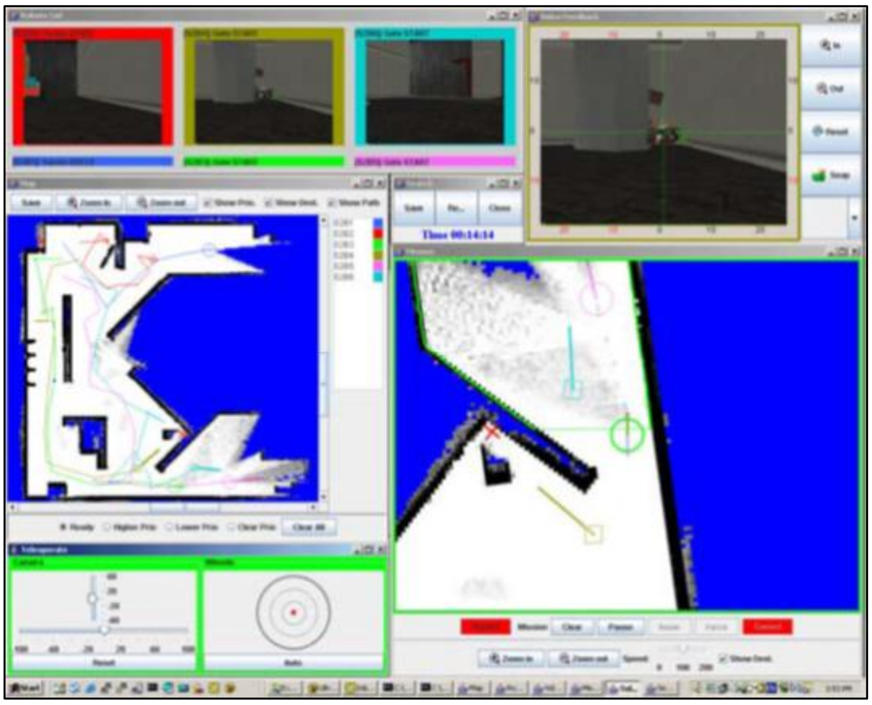


Figure 1.2 A multi-robot interface developed by the University of Pittsburgh and Carnegie Mellon University. Four camera feeds are shown and two maps. The system was developed and tested using simulated robot data. (Wang, et al., 2008).

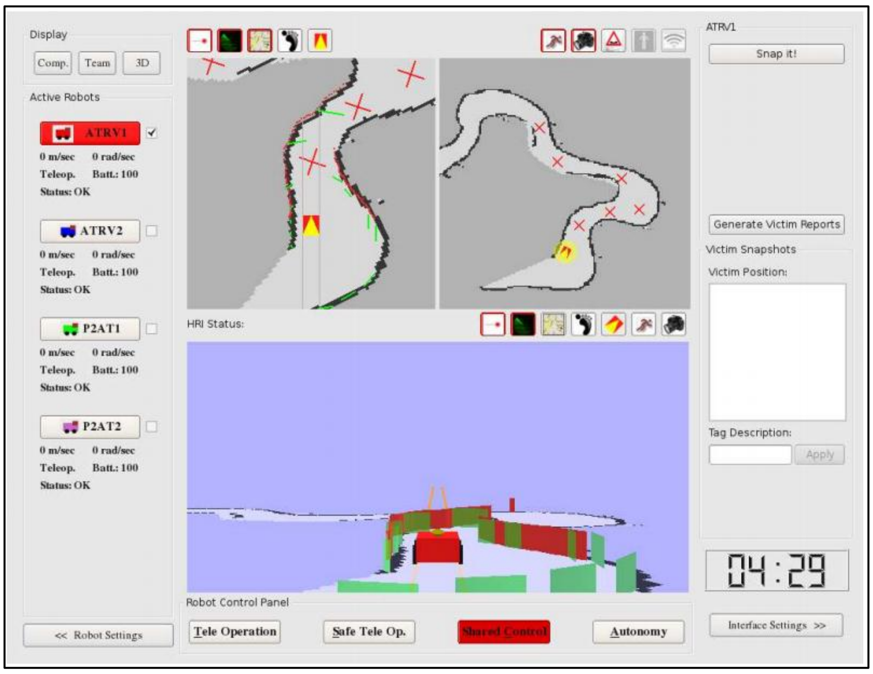


Figure 1.3 A multi user, multi robot interface developed in 2010 by the University of Rome La Sapienza. Demonstrated in the screenshot is a user controlling a single robot by setting multiple waypoints (Gomez, 2010).

The MAGIC2010 completion produced many state of the art multi-robot systems each having an advanced user interface. Notable interfaces were produced by Team Michigan (Olson, et al., 2012) and RASR (Lacaze, et al., 2012). It should be noted that all the interfaces used were designed to be used by two operators over multiple screens.

The RASR team placed third in the completion. The user interface drew design ideas from RTS games with a design philosophy of providing operators with high level system controls. The interface was distributed over two screens: the main map-centric display allowed users to set waypoints and displayed basic robot status information, shown in Figure 1.4, while the second screen was used to show the robot video feed (Lacaze, et al., 2012).

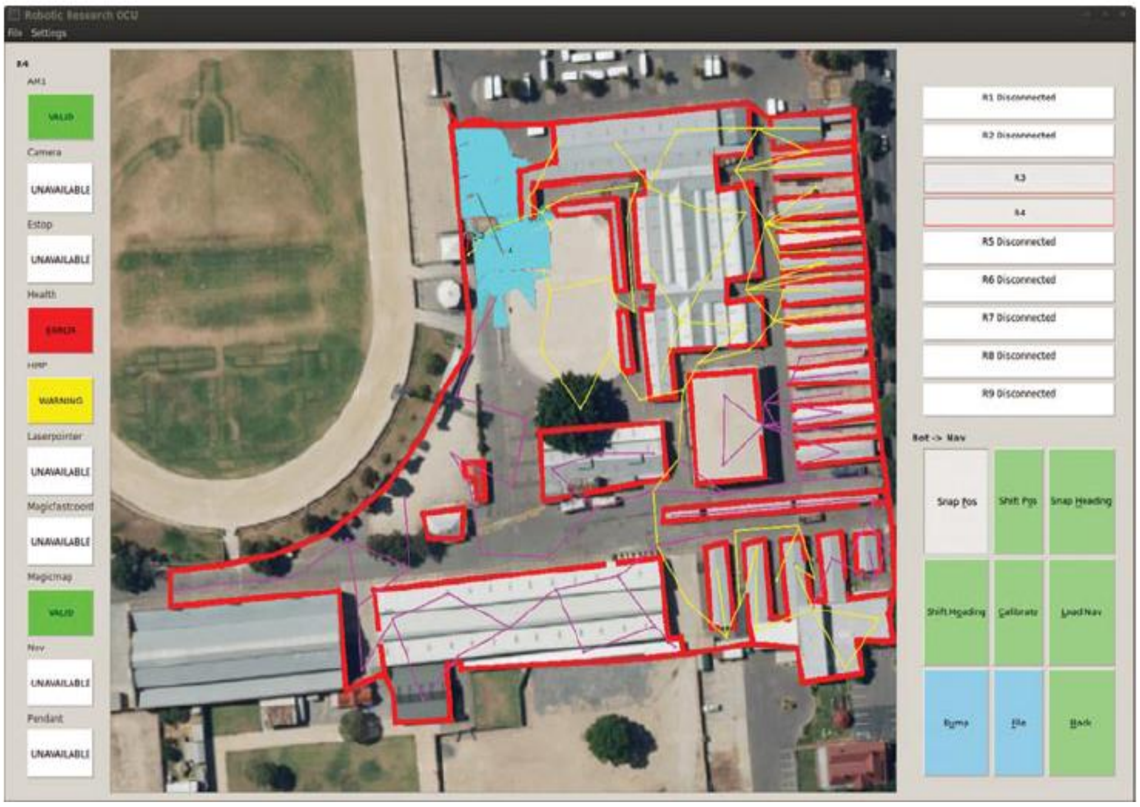


Figure 1.4 A screenshot from the system developed by the RASR team for MAGIC2010. Shown on the right hand side is the robot status feedback mechanism. The central display shows the robots exploring the environment with overlayed robot position (Lacaze, et al., 2012).

Team Michigan placed first in MAGIC2010. Their user interface consisted of four screens each displayed on a separated computer monitor requiring two users for control. The four screens were defined as: a sensor operator console which was used for object identification and to display sensor information; a task operator console, used to send waypoint to the robots; a status dashboard which displays the status of each robot and a Situation, Actions, Goals, Environment (SAGE) Display used to display the overall status of the mission (Olson, et al., 2012). Although this is a state of the art the system it still requires two operators for effective control. The team commented that further work was required to optimise human operators into the system (Crossman, et al., 2012). The system was tailored for users with high robotic experience and thus will be unsuitable for a single inexperienced user to achieve effective control. The Team Michigan interface will be considered the most state of the art of all systems analysed. Figure 1.5 shows a set of screenshots of the Team Michigan system.

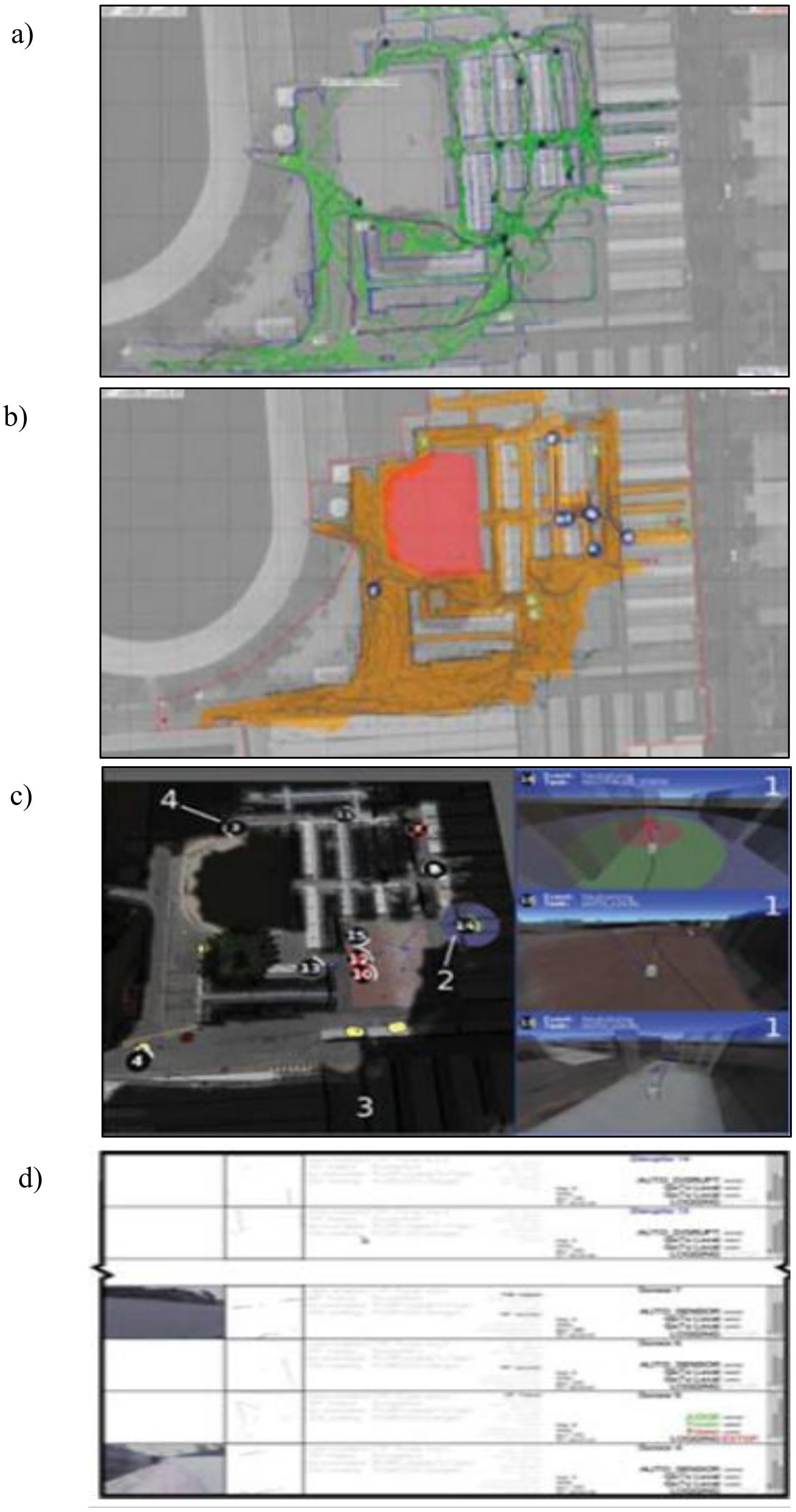


Figure 1.5 A set of screenshots showing the full user interface developed by Team Michigan a) Sensor Operator console, b) Task Operator console c) SAGE Display d) Status Dashboard (Olson, et al., 2012).

2. Process

2.1 Design Constraints

2.1.1 Hardware

The robotic fleet for the project was based off the existing the MAGIC2010 hardware; with the fleet consisting of five robots .The setup, as shown in Figure 2.1, was built on a Pioneer 3AT four wheel differential base. Each robot had an automotive core-2 duo PC running Ubuntu with Wi-Fi connectivity. Two laser ranger finders where included, the primary laser being a SICK100 with a range of 20 m, a 270 degree field of view and a scanning frequency of 50Hz. The secondary laser was a Hokuyo URG, with a range of 4m and a field of view of 240 degrees, angled down ward to detect low lying objects. Each robot was also equipped with an Xsens MTi Inertial Measurement Unit (IMU), a set of speakers and a webcam used for teleoperation and object detection.

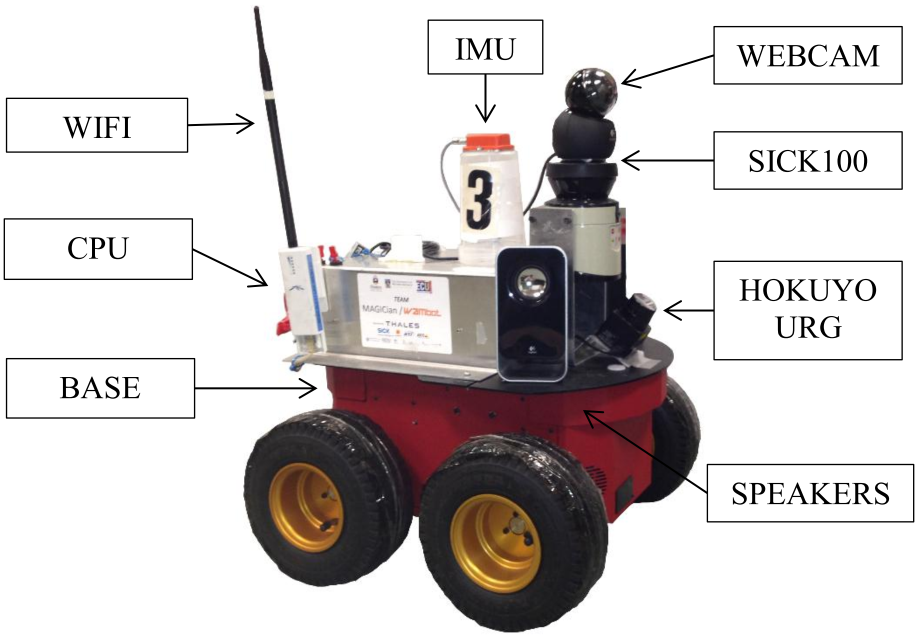


Figure 2.1 A image illustrating the hardware components of each robot.

2.1.2 Safety

Safety is an important consideration when dealing with any robotic system. Even with the inclusion of a robust and well tested collision avoidance system precautions were taken to ensure no bystander was ever hurt during system testing. All preliminary testing was conducted within a controlled environment, room 3.13 of the Electrical Engineering Building. For tests that required inexperienced users or a larger environment the robot speed was reduced. In situations were safe operation could not be irrefutably confirmed an experienced operator followed the robots and had the ability to disable the system if a problem arose.

2.2 Tools

2.2.1 Ubuntu

All code for the project was developed for the Ubuntu operating system. Although ROS is supported for Windows and OS X the builds are experimental and it is highly recommended by the ROS team to develop on Ubuntu (Willow Garage, 2012).

2.2.2 Qt

Qt is a cross platform application framework used to develop user interface layouts (Qt Project, 2013). Many open source interface frameworks are available but Qt was chosen due to its high compatibility with ROS (Willow Garage, 2012), large community and the long list of released programs developed including Skype, Mathematica and Autodesk Maya (Qt Project, 2013).

2.2.3 Programing Language

Development of ROS systems can be done in either Python or C++. For this project the majority of the programing was written in Python. Python is a general purpose high level programming language that comes highly recommended by the creators of ROS for user interface development (Thomas, et al., 2013). Python is well suited to prototyping projects, is highly readable and the growing trend in the ROS community is leaning towards systems written in the language. Where speed is absolutely essential components were written in C++.

2.2.4 Photoshop

All icon design and visual features were created in Adobe Photoshop CS6.

2.3 Design Guidelines

Modern HCI development procedures recommend developing a set of design guide lines when designing complex systems (Sharp, et al., 2011). This helps keep the product on course during development and can be used for system evaluation. The following nine guidelines were created for the project.

2.3.1 Agile Development

An agile design process was applied, by developing the project thought iterative design cycles. At the end of each cycle the product was evaluated and appropriate changes were made.

2.3.2 User Experience Centred Design

To create highly usable systems, users much be involved in all stages of development (Sharp, et al., 2011). The target user group was defined to be users with low levels of robotic experience with relatively high levels of computer competency. To enable frequent user interaction a user base was created containing four users with varying levels of robotic experience. Feedback was gathered from these four users at frequent intervals and was taken into account in all development phases. The users included:

1) Thomas Bräunl, the project supervisor who is highly experienced in robotics and whose role was to ensure the project stays within the outline of the project requirements.
2) Chris Parkin, a computer game developer and graduate mechanical engineer.
3) Prawi Woods an experienced StarCraft player and a student civil engineer.
4) Calum Meiklejoh, an experienced ROS user and graduate robotics engineer

2.3.3 Video Game Inspired Design

The interface was inspired by modern RTS computer games. Critical features were constructed to be similar to modern popular computer games including replication of the mouse and keyboard control structure, and the visual layout. Two RTS games were used for analysis, StarCraft 2 and Company of Heroes.

2.3.4 Modular Design

Modular software construction principles were applied using an object orientated paradigm and the use of the ROS node architecture.

2.3.5 ROS Compatible

The system will use ROS components where possible. The final system will be made open source and released to the ROS community upon completion.

2.3.6 Scalable

A user should be able to control five robots with the same relative ease and effectiveness as controlling one robot.

2.3.7 Reduce Cognitive Load

The cognitive load on the user was minimised wherever possible. The aesthetic design of the interface was kept clean and simple.

2.3.8 Enhance Situational Awareness

The interface prioritised increasing all forms of situational awareness.

2.4 Development Cycles

The project development plan had four main cycles.

1) Research - develop a firm understanding of state of the art multi-robot interfaces, the theories needed to develop one and learn the necessary tools required to construct such a system.
2) Design - create a well thought out state of the art system design through the use of low fidelity prototyping. Undertake feasibility studies to identify if the proposed system is possible and if it can be constructed within the allocated time frame.
3) Construction - build the system complying with the design constraints and conduct frequent user testing to ensure a stable and usable design is being produced.
4) Evaluation – conduct testing to determine the systems stability and usability.

2.5 Requirement Analysis

Project requirements were defined as the high level system critical components and the low level feature requirements that would follow the design guidelines. The requirements for the system were keep flexible to ensure an agile development. The program was developed in cycles with continuous evaluation used to further refine the system. If testing uncovered new requirements or found current requirements to be obsolete the system would be changed. The process emphasised keeping the development dynamic.

A base set of core requirements were defined in an early production cycle through research into HCI, HRI, RTS games and current robotic systems. This coupled with early prototype systems and user involvement, particularly involvement from Professor Bräunl, produced a base set of requirements as outlined below.

2.5.1 System Critical Components

1) Selection Method - a simple method to select a robot.
2) Simple Control System - a simple method to control the robots with limited human interaction.
3) Full World Overview, increased location awareness by having a clear view of the world positioned from a bird's eye view.
4) Full System Overview - users should know at all times the status of all robots, increasing the users overall mission awareness.
5) Status Identification **-** users need to be able to identify the exact status of a robot, increasing status awareness. The solution should require limited cognitive load.

2.5.2 System Feature Requirements

1) Display the robot generated maps and allows users to easily navigate them.
2) Stream video from robots.
3) Multiple control methods; waypoint, command buttons and a controller.
4) Display status of each robot.
5) High level of robot autonomy.
6) An Object detection system.
7) Feedback mechanisms.
8) A robot monitoring system to increase system robustness.
9) Robust communication protocol.

2.6 Evaluation

2.6.1 Agile Evaluation

In keeping with the agile development mentality, evaluation methods were continuously applied throughout the design and construction phases. Frequent testing was used to both improve the initial requirement list and refine the system. At regular intervals evaluation sessions were conducted in which the current system will be compared against the design guidelines, changes made were noted. User tests were also conducted at regular intervals using the user base. Tests were performed in an informal setting and involved showing the users the system to get feedback.

2.6.2 Final System Evaluation

Final system evaluation was done in two parts, system stability testing and system usability testing. Stability testing involved creating test cases and simulating extreme cases for the system to complete. Usability was tested by user tests, which involved users completing tasks with the system. Results from all tests were then used to complete a heuristic evaluation of the whole system.

2.6.3 Stability Testing

Stability testing of the system was performed to ensure both stability and robustness. Test cases were designed to test all system features and components in both general and extreme cases. Tests had a pass/fail outcome with notable results being recorded. All bugs found were recorded in a bug tracker and prioritised depending on their severity.

2.6.4 Usability Tests

The final user tests were centred on evaluating the systems usability. The main goal was to prove the interface is an effective means for a single user to control a robotic fleet. Users were chosen to have varying backgrounds in robotics and pilot tests were conducted prior to the main testing to ensure the testing procedure was adequate. The format of each test can be summarised as follows:

1) Introduction and User Background identification.
2) Task Identification.
3) Single Robot Test.
4) Questioning.
5) Multi Robot Test.
6) Questioning.

An introduction was used to gather information about the user's background and identify the users experience level in both robotic and real time strategy gaming. A rank between zero and ten was assigned to each user, with ten implying an expert level of experience. The task identification test involved users completing simple system tasks as listed below, with users given no instruction on how to use the system only being told it was based off an RTS game. The time taken to complete the task as well as the number of wrong actions taken where recorded. An action was defined as issuing a command that could influence the system. If a user asked for assistance it was given and recorded. The tasks were;

1) Move the camera.
2) Zoom the camera in and out.
3) Select a robot.

4) Rotate a robot.
5) Determine the battery level of robot one.
6) Determine the status of the camera of robot two.
7) Determine out the error in robot one.
8) Send robot two a goal.
9) Stop robot one.
10) Determine he real time obstacles surrounding robot one.

The funcation of task identification tests were to investigate how well users can figure out the appropriate actions needed to perform a task given they have a limited understanding of the system. A recording of two robots was used to implement the test to ensure the test conditions were kept constant. The test had a secondary purpose of determining if a user was skilled enough to control the system safely.

Following the task identification users were given control of a single robot to complete a task completion test. An instructional brief was given to ensure the user completely understood how to control the system. Users were then asked to drive the robot to a destination, first using teleoperation (a video feed and controller), followed by using the developed interface. The time to destination was recorded as well as any potential robot crashes. An experienced operator was present during all tests to follow the robot and stop the system if the user was in danger of harming a bystander or damaging the robot. Half of the users were asked to complete the tests in the opposite order (i.e. teleoperation after interface operation) to negate the learning effect on the second test.

A multi-robot task completion test was undertaken last. Users were given control of two robots and were asked to map the third floor of the Electrical Engineering building. Time to complete the task and potential crashes were recorded.

Questioning was performed after each test using an informal interview structure that attempted to start conversations with the user. Feedback was recorded to help draw conclusions about the system usability and to investigate possible improvements or features.

2.6.5 Heuristic Evaluation

A heuristic evaluation was performed based on the conclusion of all results collected

The criteria used were developed by the Open Source Robotics Foundation and is specially designed for robotic interfaces (Steffi, 2013). Items were ranked between zero and four, as shown in table 2.1.

Score	**Rank**
0	Not a problem
1	Cosmetic problem only
2	Minor usability problem
3	Major usability problem
4	Usability catastrophe

Table 2.1 The ranking system used for the heuristic evaluation.

When rating an item the following will also be taken into account. Frequency, how often the issue happens, impact, how hard is it to recover from the issue and persistence, will user have the issue over and over again.

2.7 Robot Software Design

To facilitate a working robotic interface a substantial amount of software had to be implemented on each robot. In order to save time, increase the overall system quality and increase the ROS compatibility most of this software was implemented using open source ROS packages.

2.7.1 Hardware drivers

Each robot uses a number of sensors that require drives; these were implemented thought ROS packages and set up via ROS nodes. When selecting driver packages forums where reviewed and basic tests were carried out to ensure the package was stable. The SICK1000 laser range scanner used the package LMSxx (Banachowicz, 2011) and the Hokyou URG laser scanner used the hokuyo node package (Gerkey, et al., 2012). Both drivers were chosen for having good reviews. The base drivers were implemented using the p2os_driver package (Feil-Seifer, 2012). This package was chosen due to it having good troubleshooting resources, diagnostics features, good user comments and a high level of refinement.

2.7.2 Mapping and Robot Localisation

For map generation and robot localisation the ROS package hector_slam was used (Kohlbrecher & Meyer, 2013). The package uses Simultaneous Localisation and Mapping (SLAM). This allows the robot to build a map of an unknown environment while simultaneously locating the robots position within that map. By using laser scan data, robot odometry and thought the use of an Extended Kalman Filter detailed maps of the environment can be created down to a resolution of 5cm (Kohlbrecher & Meyer, 2013) . The robots position can then be localised with in this map.

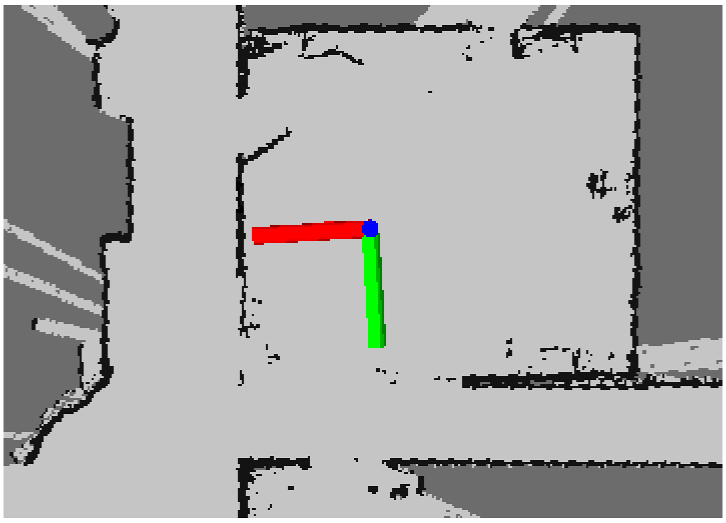


Figure 2.2 A robot using SLAM to map an environment with the boundaries shown in black and free space shown in grey. The localised robot position is shown in the centre, displayed in red and green. (Kohlbrecher, 2012).

2.7.3 Navigation and Collision Avoidance

To facilitate navigation and collision avoidance each robot runs the ROS Navigation Stack (Marder-Eppstein, 2013). Using this robot can plan paths to a goal while avoiding obstacles. The local map created by hector_slam is used to generate a costmap, where by a high cost relates to an obstacle. The path of lowest cost will be the path avoiding all obstacles. The navigation stack was one of the first packages released for ROS and was primarily designed by Willow Garage, the creators of ROS. This makes the package highly compatible with ROS and very refined.

The primary laser scanner is placed to take horizontal scans to maximise its range. This however will not detect low lying features which could potentially be hazardous to the robots. To detect low lying obstacles the Hokuyo laser scanner was angled downwards. Any low lying objects will intersect with the laser and can be added to the cost map.

2.7.3 Communication

To allow a single master computer to communicate with multiple robots a multi-master communication system is required. This will facilitate wireless data transfer between a master computer and a number of robots. The ROS Multimaster Special Interest Group provides a summary of all current multi-master packages available on ROS (Stonier, 2013). Thought analysis of existing techniques a decision was made to use the ROS package multimaster (Meeussen, 2011). Data is transmitted using the ROS package foreign relay (Gassend, 2012), which permits a topic on one computer to be published on a second remote computer. The multimaster package uses a configuration file defined on the master computer to define which topics the master will send and receive. This makes the system much simpler and requires only one node to be run for each robot.

2.8 Interface Design

Primary interface features went thought a process of iterative refinement using the iterative procedure defined in early sections.

2.8.1 Central Display

The implementation of the central map display had two design options, a graphical environment using OpenGL (Open Graphics Language) or a 2D image environment using Qt Image features. The image solution would be simpler and have a faster development time while the graphical solution would have more flexibility and in future versions could be made into a 3D system. It was decided to use a graphical environment implementing using a QGLWidget (Qt, 2013). Using a Qt implementation of OpenGL provides a simpler and faster development process when compared to a stranded OpenGL system. Maps can be rendered using textures with GPU acceleration producing a fast and seamless map update system. QGLWidget's also contain an effective screen to world coordinates transformation mechanism, which was sued to implement the robot selection and waypoint features.

2.8.2 RQT

Initially the program was designed to be a standalone product, however towards the end of the construction phase ROS released the rqt framework. This had profound influence on the course of the interface design. Rqt is a Qt-based framework for ROS GUI development; it provides a base to which plugins can be run. The system functions by having 'dockable' modules that can that be moved and resized. Available plugins include; Image View which is used to display video feeds, Bag, a system for recording and playing back ROS topics, Launch that can be used to launch ROS launch files in a user friendly manner, Shell which is an embedded terminal and many more.

Using the rqt framework a dynamic and module interface could be made. Modules can then be selected by the user depending on the task at hand. Furthermore developing with regards to rqt conventions and standards will result in a high ROS compatible interface. The decision was made to re- design various interface components to allow for rqt compatibility.

2.8.3 Interface Visual Design

The interface visual design had five main revisions; these can be seen in Appendix A. The final design was based preliminary off identified RTS features. Key components included a main map and a control panel consisting of a mini map, control and a display panel to display robot data. Unlike a video game a robotic interface requires the user have more control over the system. To this end a series of system controls where added to facilitate full system control options. To increase overall mission awareness a robot list was added which provides a user with a state summary of each robot. A message bar was also added promote user feedback. The visual design was designed to be clean and simple with clear boundaries and an appropriate colour pallet all to help manage user attention. A video feed will be available thought the Image View rqt plugin. A low fidelity prototype of the interface layout can be seen in figure 2.3.

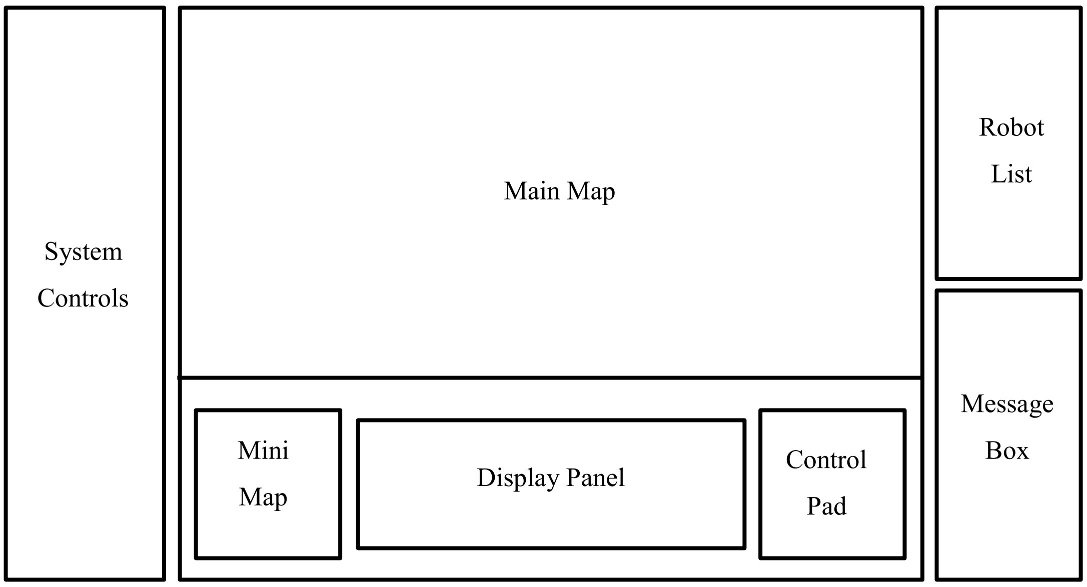


Figure 2.3 Final interface layout with the main map and control panel positioned to be the central display features.

Many possible plugin combinations can be formed using RQT, the user can customise the system depending on the mission. If for example a constant video feed for all robots is required, then multiple instances of the Image View plugin can be run.

2.8.4 Feedback Mechanisms

To ensure the user has the optimal view of the interfaces current state, feedback mechanisms were used. The message bar provides users with important system information with a sound played upon creation of a new message. Sound was used in other situations to provide feedback. When a unit is selected or a goal is transmitted unique sounds will be played. Hints were overlayed on icons and buttons being activated when a user hovers over them. These are implanted through Qt's tooltip function.

2.8.5 Icons

Proper icon design is a critical part of refining any user interface. The icons used in the system where based upon open source images obtained from (Oxygen Team, 2011) which are free for non-commercial use. Icons where then modified in Photoshop to provide the correct design and to comply with the ascetic style of the interface. The remainder of the icons where created in Photoshop.

2.9 User Interaction

User interaction was specifically designed with the main goals of; minimising the cognitive load on the user, increasing situational awareness and to provide a simple and easy to use experience for the user.

2.9.1 Robot State Summary and Status

The supply the user with continuous feedback about the overall state of each robot a robot state summary was displayed using the robot list. The list shows the names of each robot with an adjacent icon, this icon represents the overall state of the robot. The feature will provide the user with an overview of all the robots, helping to increase the users overall mission and status awareness.

2.9.2 Increasing Situational Awareness

Increasing the situational awareness was defined as a high priority in the design guidelines. However by increase situational awareness more data is needed to be displayed this in turn increases the cognitive load on the user. Thus a trade-off can occur. To ensure a fine balance is met continuous testing on a wide variety of users was undertaken. The amount of data a user can handle will mainly depend of the experience of the user and the current situation. To keep the system flexible data features will have the ability to be hidden. Situational awareness was further increased with the following features:

1) Overall map with localised robot position, increases location awareness and surroundings awareness.
2) Robot icon with direction, increases location awareness
3) Cost maps , increases surroundings awareness

Colour coding was applied by setting the robot as a base colour with the map and costmaps as darker and lighter shades of the base colour respectively.

2.10 System Design

2.10.1 Reducing Network Traffic

The amount of data each robot sends over the Wi-Fi network is critical to the systems scalability. If too much data is sent over the network the ROS system will begin to suffer from lag. Table 2.2 shows the different bandwidth usage of each message.

Demonstrating that 88.4% of network bandwidth is due to the sending of the map and 9% is due to the video feed. To reduce the network load both the map and video feed where compressed.

Type	**Band width**	**Percentage**
Map	2250 Kb/s	88.4%
Costmap	50 Kb/s	1.96%
Path	10 Kb/s	0.4%
Pose	4 Kb/s	0.15%
Image	230 Kb/s	9%

Table 2.2 The calculated bandwidth for various ROS topics, bandwidth is measured in kilobytes per second. With the percentage of the message bandwidth relative to the total transmitted bandwidth is also displayed.

2.10.2 Global Origin

A critical aspect for a state of the art multi-robot system is for the system to share a common global origin. When a user selects a point on the map a global goal is being created, the robot must then be able to translate this to its local origin and thus move to that position. For this to be feasible the robot must know its own position within a global frame of reference. The original project plan involved using a software module called Map Builder. Developed by Rob Reid for the MAGIC 2010 completion the program uses a gird based SLAM that is capable of producing 500 x 500 meter maps from sub maps supplied by the robots (Reid & Braunl, 2011). The smaller sub maps are fused on the master computer.

At the time of the completion of this project Map Builder was still in the process of being ported to ROS by a fellow researcher. The system however was not ready to be integrated into a user interface with testing showing the system would break down with an increasing number of robots, an example of this is shown in Figure 2.4.

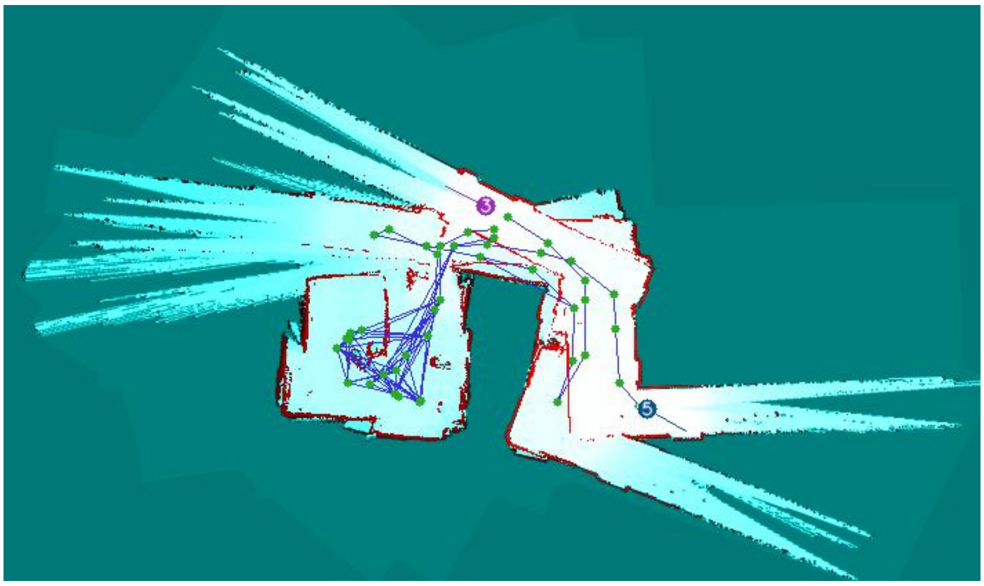


Figure 2.4 A screenshot showing Map Building failing to fuse two robot maps.

A global origin was still required to implement a multi-robot system. A temporary solution involved starting the robots in the same location and orientation resulting in the local maps from each robot roughly lining up. By then creating effective user interface features a user was able to effectively manage the system. Future work will focus on completing the Map Builder port and integrating it into the interface.

3. Final Design

The final interface design is composed to two main systems; the master base station and the robots. The master is comprised of the graphical user interface, run on a standard computer. The robots each run software locally with bi-directional communication facilitated between the master and the robots.

The system created displays an environmental map with localised robot positions. A single user can command multiple robots thought the use of waypoints and receive efficiently organised data of each robot. An object detection system was implemented that allows robots to detect objects of interest.

For a copy of the full system software please contact Professor Thomas Bräunl.

3.1 Master Software

Figure 3.1 shows the software module block diagram of the master system. The system is designed in a hieratical fashion with the GUI Module or main class situated at the top with other features implemented as sub classes.

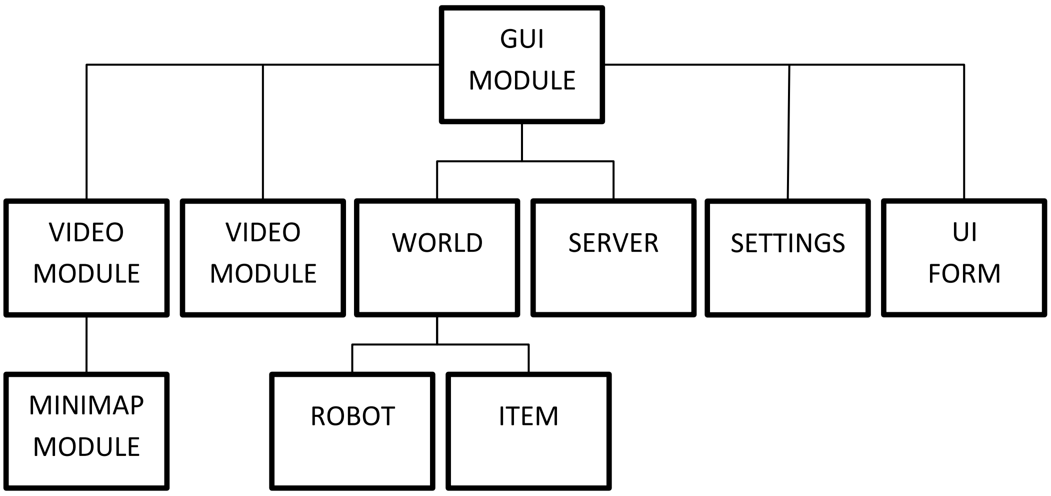


Figure 3.1 The final software block diagram for the master user interface.

The main class, named GUI module is the parent class and deals with all interface logic. This includes the function of buttons to the text labels displayed, as well as maintaining all the sub classes. Sub classes are implemented to increase system modularity which also aids makes testing easier. As a basic overview; the Map module creates the central interface display, the server class is responsible for communication with the robots; the video module enables a video feed to be displayed and the world class represents the system data structures.

3.1.1 UI Form

The form represents the skin of the interface, defining the position and visual look of each of the interfaces components. QtCreator was used to create the form for this system (Qt Project, 2013).

3.1.2 World

The world class attempts to create a virtual representation of the robots environment. This class holds all current robots and items in the environment. The data of any given robot is stored in a separate sub class which attempts to synchronise its data with its real counterpart. An item sub class is defined to represent objects of interest with in the environment.

3.1.3 Settings

The settings class enables program features to be modified thought a configuration file.

3.2 <u>Robot Software</u>

The final system running on each of the robots consisted of a combination of open source and custom developed packages. Details of the selected open source packages can be found in section 2.7.

3.2.1 Custom Packages

Custom robot software was developed including; robot comms a basic server for the robot. This class receives commands from the master server. Process Manager, a class that monitors the status of the robot. Detector, a class to manage the object detection algorithms. Name Convertor, remaps topic names to include a prefix determined by the robots name which are used to send to the master.

3.3 <u>Communication</u>

Communication between the master and robots was enabled by a Wi-Fi network using the standard IEEE 802.11 protocol and the ROS multimaster package to facilitate data transfer. The system’s design was based around each robot communicating with a single master. Outgoing data from the master will be transmitted from the server class with incoming the data from each robot being read by the respective robot classes. Each robot sends the following topic with the associated topic name to the master, as shown in Table 3.1.

Description	**ROS Message Type**	**Target Topic Name**
Current position	geometry_msgs/PoseStamped	/pose
Local map	nav_msgs/OccupancyGrid	/map
Costmap	nav_msgs/GridCells	/costmap
Path to planned goal	nav_msgs/Path	/path
Video feed	sensor_msgs/CompressedImage	/video_feed

Table 3.1 A list of the data messages each robot transits to the master computer. Using the Name Convertor class these topics are renamed to start with a prefix determined by the robots name, for example "wambot4/path".

A low priority communication protocol (LPCP) was implemented to allow simple messages of low frequency or bandwidth to be transferred. The system was designed to transmit messages that include robot goals, robot state changes and objects of interest that robots have detected. An overview of the messages used in LPCP is shown in Table 3.2, for further details of these messages see Appendix B.

Message Name	**Purpose**
robot_to_base	Transmit robot data to the master
base_to_robot	Transmit master data to a robot
ObjectOfIntrest	Contains the details of an object detected by the robot. Sent from robot to the master

Table 3.2 Overview of the custom messages used in LPCP.

The master receives and transmits a unique topic for each robot. All robots publish to the same master topic with a name tag contained within the message to identify its sender to the master. The communication is illustrated in Figure 3.2.

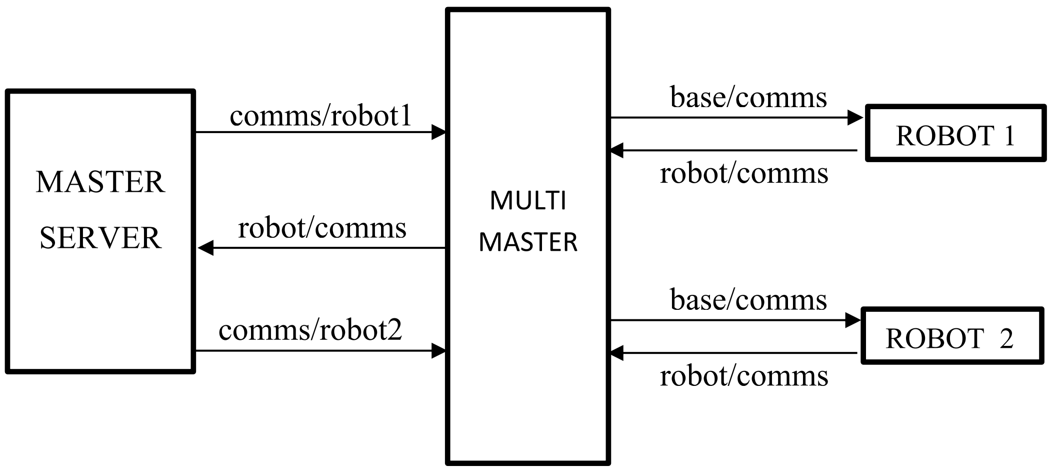


Figure 3.2 LPCP block diagram, illustrating communication between a single master and multiple robots. The topic names are displayed.

3.4 Robot State

Each robot maintains an internal state that defines the status of its hardware, battery level and current operational modes. The state is defined by nine components as shown below:

1) Base status.
2) Primary laser status.
3) Secondary laser status.
4) Camera status.
5) IMU status.
6) Connection status.
7) Control mode, refers the robots control system. For example manual verses autonomous exploration.
8) Detect mode, refers to what objects the robot is looking for.
9) Failsafe mode, refers to the actions the robot takes if it loses connection with the base.

The six hardware status features including the base, lasers, camera, IMU and connection are all monitored by a Process Manager node which is run locally on each robot. The advantage of doing management locally is that if connection is lost, the system will continue to make informed decisions independently of a master computer.

All hardware features are primarily monitored using a data flow checker. The hardware's data topic is subscribed too, if the streaming of data stops the hardware is said to fail. A middle warning state can be determined for the base and secondary laser by examining the diagnostics topic outputted by the base drivers. The robot will update the master of its current state using the LPCP. An array is sent containing the current state as shown in Table 3.3.

Index	**Item**	**Representation**
1	Base status	0 to 2
2	Primary Laser Status	0 to 2
3	Secondary Laser Status	0 to 2
4	Camera	0 to 2
5	IMU	0 to 2
6	Battery Level	Percentage
7	Current Mode	0 to number of modes
8	Current Detect Mode	0 to number of modes
9	Current Failsafe	0 to number of modes

Table 3.3 The array sent by each robot which defines its state is made up of nice components.

3.5 Master Communication

Using the LPCP a robots state can also be modified by using the base_to_robot message. Three main state modifiers are included in the system; mode, object detection and failsafe. All state changes can be made from the use of a Qt QCombox, as shown in Figure 3.3.

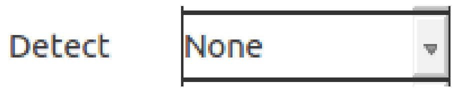
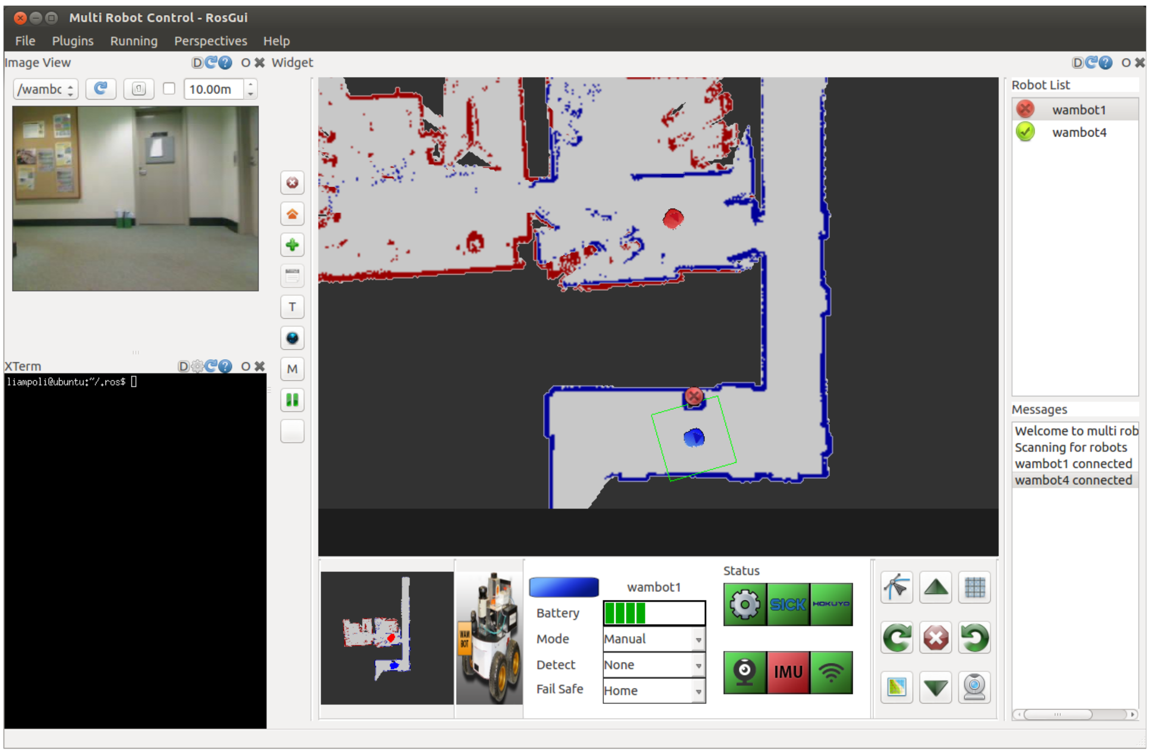


Figure 3.3 A example of a QComboBox used to change the robots object detection algorithm. Currently the robot is set to detect no objects.

The mode refers to the control state the robot is in this includes two sub states; manual control and autonomous exploration. Manual control lets the user control the selected robot via waypoints or by using a controller. In the explore mode, the robot will explore autonomously. The detect mode enables to user to change the current object detection algorithm run on the robot.

The failsafe state is used to determine what actions the robot will take if connection is lost with the master. If the robot loses connection it can take one of three actions; return home (defined as the place the robot was turned on), stop or do nothing. If connection is lost due to the robot going out of network range the home option will return the robot to within range.

3.6 Full System



```
liampoli@ubuntu:~/.ros$
```

Robot List: wambot1, wambot4

wambot1
Battery
Mode: Manual
Detect: None
Fail Safe: Home

Messages:
Welcome to multi rob
Scanning for robots
wambot1 connected
wambot4 connected

Figure 3.4 The final graphical user interface displayed on the master computer. In the screen shot the user is controlling two robots, with robot 1, shown in blue currently selected.

3.6.1 Map Module

The map module implements the central display and is capable of displaying the map, robot position, robot paths, costmaps and detected items. The module is implanted using a QGLWidget, using an orthogonal camera view; positioned form a bird's eye view of the environment. The final image displayed to the user is a layering of many parts. By using the z-axis feature of OpenGL components can be painted at different height levels. Thus when viewed from above the image will appear as a single image. The layering order is: maps, costmaps, robot positions, robot paths, items then robot status icons. The map presents a relatively large data object that needs to be updated frequently, thus an efficient technique of updating and displaying the maps is required. The mapping pipeline involves a number of steps to get from a robot generated map to an image displayed to the user.

Robot maps are first converted to a ROS Image using the ROS package hector_compressed_map_transport (Kohlbrecher & Meyer, 2013). These maps are then sent over the network to the master computer. Once received the image is uncompressed using the image_transport package, running a republish node (Mihelich & Bowman, 2013). The interface reads in this image as a ROS Image message. To display the map in a Qt OpenGL environment the image must be converted to a QImage. To achieve this the ROS Image is first converted to a openCV image using the cv_bridge package (Mihelich & Bowman, 2010). From this the openCV image can be converted to a Qt QImage using Qt based functions. This QImage can then be mapped to a texture and displayed in the graphical environment.

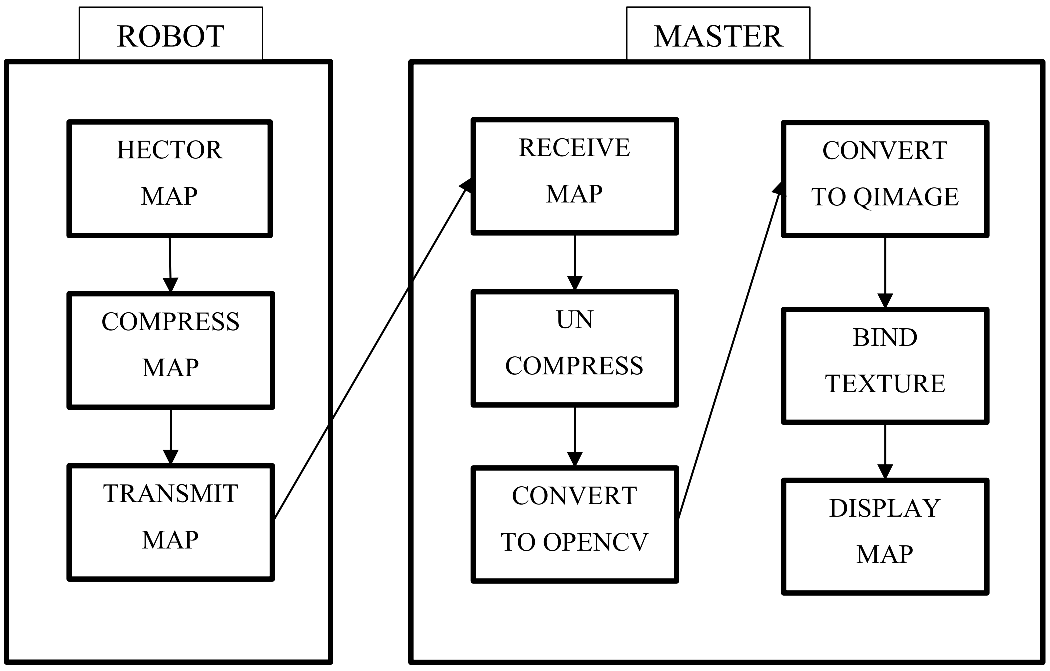


Figure 3.5 Block diagram illustrating the map display pipeline. Data transmission between the robot and master is facilitated thought the Wi-Fi network.

The robot position is displayed by using a preload icon image that is textured mapped and placed at the robots position. Cost maps and robot paths are drawn using OpenGL primitives types. Simple squares are used for the cost maps and lines to draw the paths. To help manage user attention each robots map is coloured coded efficiently using QImage colour tables. The final product is shown in Figure 3.6.

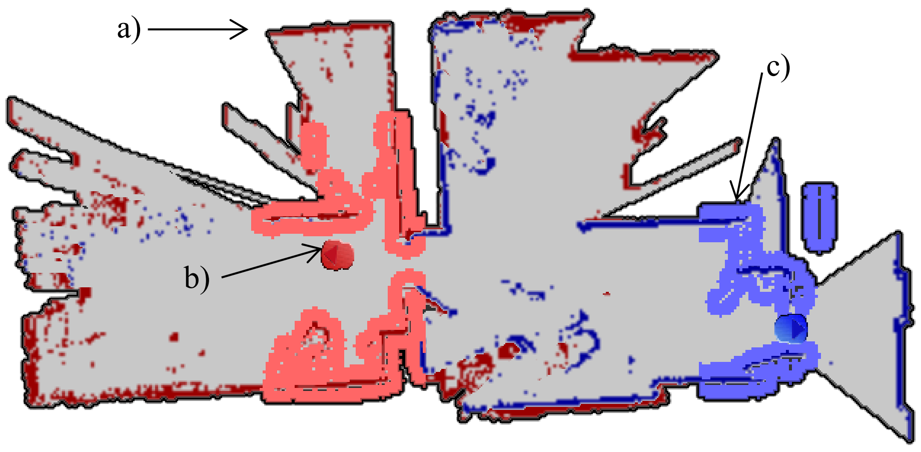


Figure 3.6 The Map Module demonstrating a) colour coded maps, b) robot position and c) colour coded costmaps.

Simple mouse controls where added to help the user navigate the map. Holding the left mouse button gave the user the ability to drag the view and the scroll bar can be to zoom in and out.

3.6.2 Navigation

The navigation features of the system allow a user to select a robot by clicking the robot icon on the map. From this the user can select a point of the map. This point will now become the robots goal. The robot will proceed to calculate a path to the goal then drive to that position. Figure 3.7 illustrates this feature.

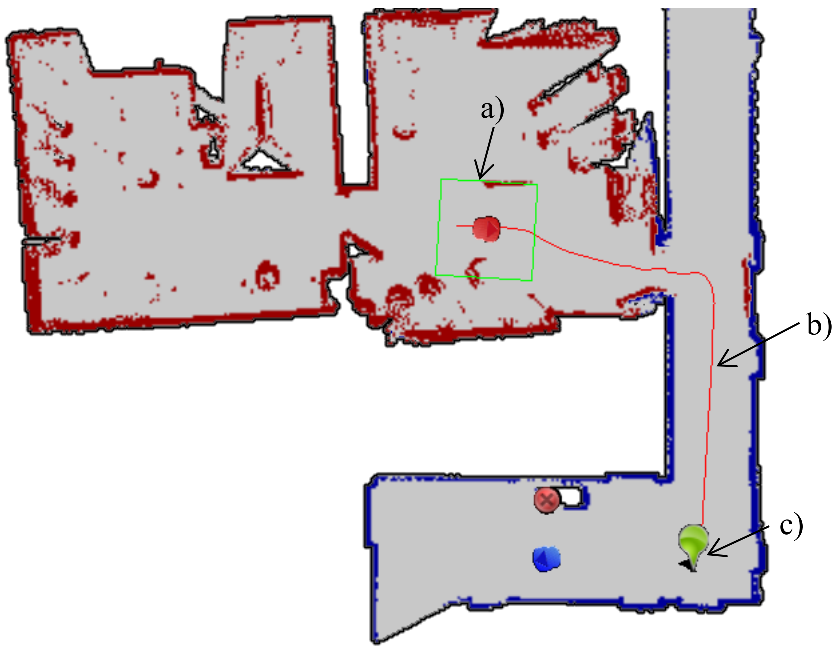


Figure 3.7 The Map Module demonstrating the navigation features. The robot is traveling along a path to a user defined goal. a) robot is selected, b) calculated path is displayed c)goal icon is displayed.

3.6.3 Control Panel

The control panel represents the primary means for a user to gather information about a robot. The panel can change the data it displays depending on the type of unit selected. Figure 3.8 illustrates the control panel for a selected robot.

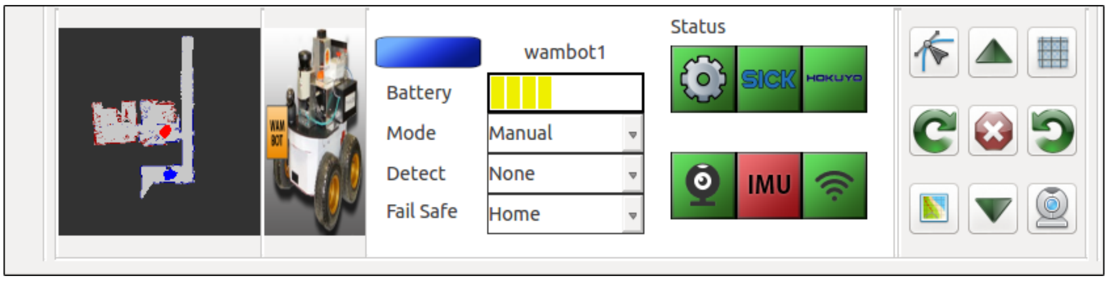


Figure 3.8 Display panel for a selected robot showing a)minimap, b)robot icon, c) central controls, d) status display and e) command pad.

Figure 3.8 shows the control panel view when a robot is selected. Figure 3.8.a shows the mini map which represents a full view of the environment with all robot positions displayed. Figure 3.8.c illustrates robot state information including the battery charge which will change colour depending on the level of charge. At over 50 precent it will be green, between 50 and 20 precent yellow and under 20 precent red. Figure 3.8.d displays the status of the robots hardware components. The base, primary laser, secondary laser, camera and IMU status are all represented by three states OK, warning and error. Represented by the colours green, yellow and red respectively shown below in Figure 3.9.

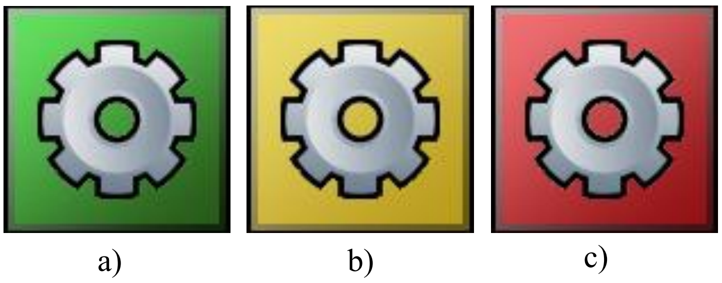


Figure 3.9 The three possible hardware states a) OK in green, b) Warning in yellow, c) Error in red.

Figure 3.8.e shows the command pad. This feature has three modes depending on which unit type is selected. If a robot is selected the button outcomes are shown below in Table 3.4.

Button	Action
Top Left	Toggle display of robot path
Top Centre	Move robot forwards
Top Right	Toggle display of robot costmap
Middle Left	Rotate robot left
Middle Centre	Stop robot
Middle Right	Rotate robot right
Bottom Left	Toggle display of map
Bottom Centre	Move robot backwards
Bottom Right	Toggle camera controls

Table 3.4 Possible controls for a selected robot.

Basic control of the robot can be performed using the control panel. Move forward and backward will send the robot a goal one meter in front or behind the robots current position. Using such a system as opposed to simply sending the robot a drive command will integrate collision avoidance. This is especially useful when commanding the robot to reverse. Each robot has a blind spot behind it as the camera and both scanners do not have sufficient field of view. The robot will only reverse if it can be certain no objects lie behind it. The toggle camera button will open the camera control pad. This can be used to move the camera.

When no unit is selected the control panel will change its display to that as shown in Figure 3.10. Using these controls the user can move the position of the camera.

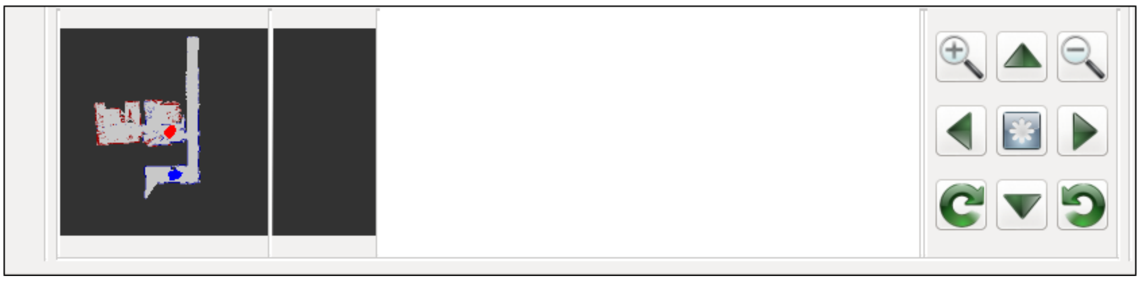


Figure 3.10 The Control Panel display when no unit is selected.

3.7 User Feedback

3.7.1 Sound

System sounds are implemented using the ROS package audio_common (Hendrix, 2012). Unique sounds are played for robot selection, goal selection and for incoming messages. All sounds used where based for open source wave files.

3.7.2 State Summary

The state summary as shown in figure 3.11.b is used to provide the user with a summary of the state of all robots in the environment. The list was designed to be interactive so that robots could be selected by selecting a list item. Hints were also added to given an explanation to the status icon. Overall five states are available with assigned priorities; only the worst case error will be shown to the user. The state summaries and there priories are shown below in Table 3.5, a priority of one is considered to be the most urgent.

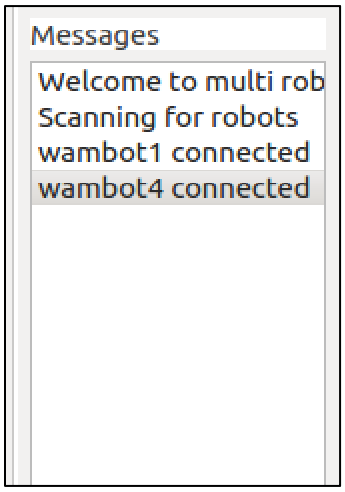
Priority	**Name**	**Icon**
1	Lost Connection	📶
2	Low Battery	⚡
3	General Error	✖
4	General Warning	!
5	No Problems	✓

Table 3.5 Possible robot state summaries, shown in descending order of priority

3.7.3 Message Box

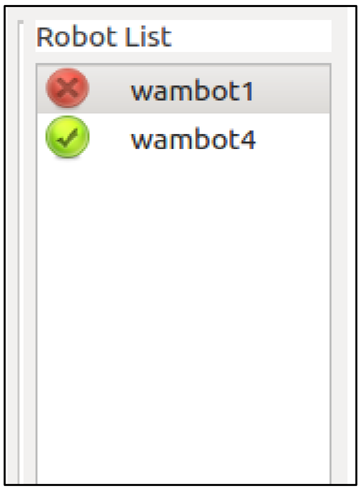
The message box as shown in figure 4.11.a is used to display system messages to the user. A QtListView is used to implement this feature.

a)


Messages

Welcome to multi rob
Scanning for robots
wambot1 connected
wambot4 connected

b)


Robot List

✗	wambot1
✓	wambot4

Figure 3.11 A screenshot showing a) Message box, b)Robot list displaying a state summary for two robots.

3.7.4 Video Feed

The video module selects which image to publish to the video_output topic. This topic can then be used by an Image View plugin. By default the module will output the video feed of the selected robot. When an item is selected the detection image will be displayed.

3.7.5 Side Panel

The side panel is located of the left side of the interface and provides the user with further system options. Table 3.6 shows the possible commands.

Button	**Action**
1	Stop all robots
2	Centre Camera – return the camera to the origin
3	Add item – a user can manual add an item to the world
4	Show summary – disabled
5	Toggle Template- Toggles showing the map template
6	Scan – scan for new robots to add to the system
7	Toggle Map control – Toggle between moving the main view or the mini map view
8	Help – Show the help page

Table 3.6 List of all possible system commands.

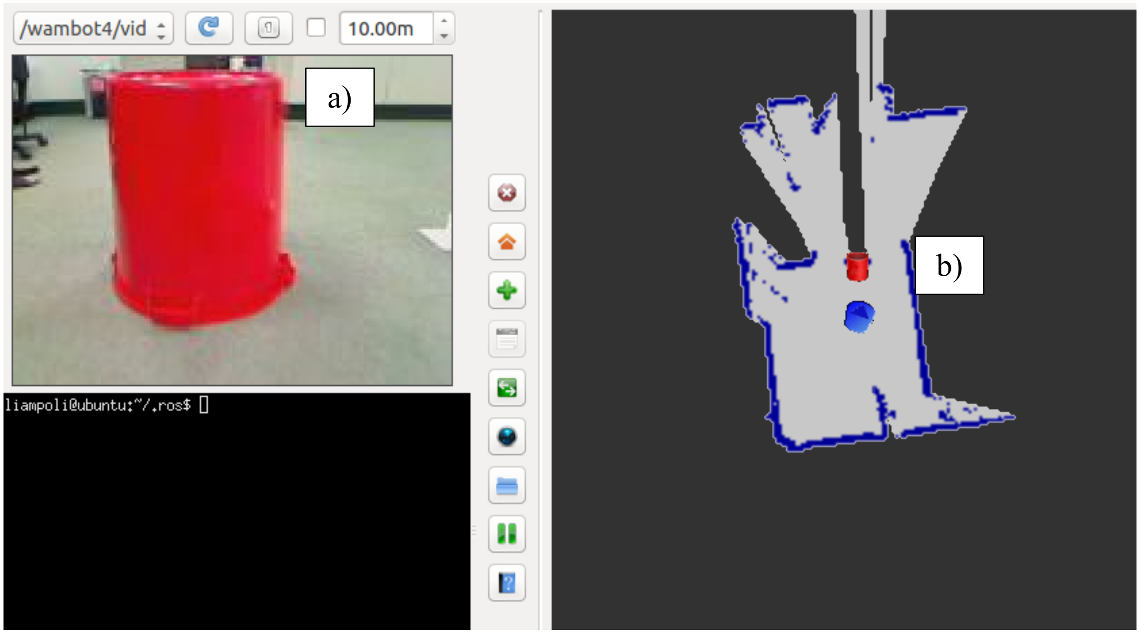
3.8 Object Detection

By integrating object detection algorithms into the system each robot can search for objects of interest independent of operator assistance. This can greatly decrease a suers the cognitive load in search based missions. Two object algorithms where created, a red barrel detector and a face detector.

The red barrel detector uses the ROS package cmvision (Lazewatsky, 2011) which uses a RGB or YUV blob detector. Cmvision outputs a list of all possible blobs to a topic, defining blob size and location. The detect node then analyses these blobs to determine if they correspond to a red barrel. The face detector algorithm was created using an openCV HarrDetector.

3.8.1 Detection Pipeline

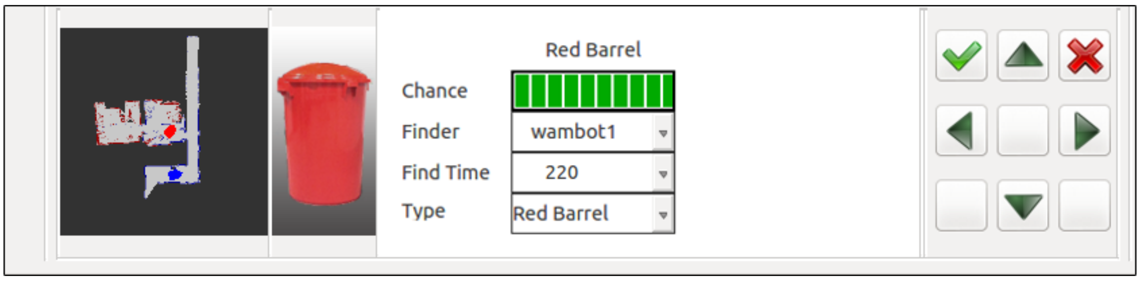
When a robot detects an object it publishes an ObjectsOfIntrest message which is read by the interface. Each message contains the robot finders name, find time, percentage chance the object is correct and the image that was used for the successful identification. The interface then displays an icon of the object on the map in the discovered location. This is shown in Figure 4.13.



```
liampoli@ubuntu:~/.ros$
```

Figure 3.12 An example of a robot detecting a red barrel a) the current robot video feed b) the red barrel icon being displayed on the map in the position of the detected item.

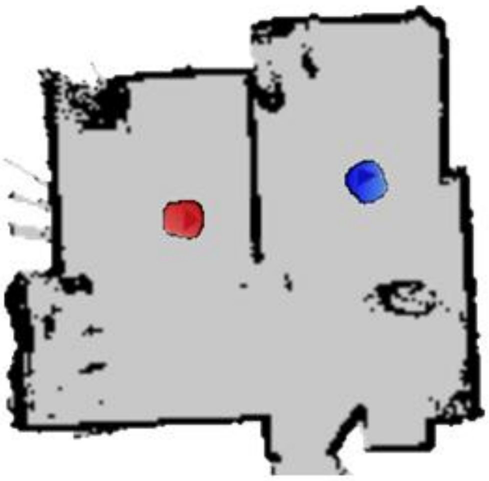
By clicking on the item the control panel will show item information and supply basic commands. The user may adjust the items position, remove it from the environment or change the items type. Figure 4.13 shows the control panel display for a selected item.



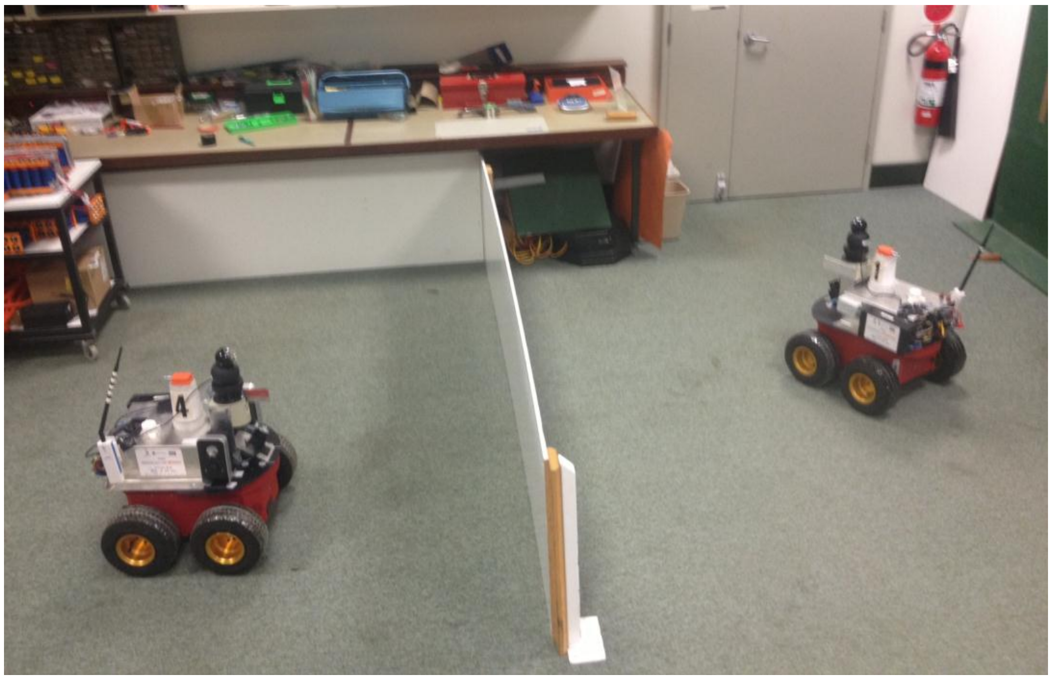
Red Barrel

Chance	
Finder	wambot1
Find Time	220
Type	Red Barrel

Figure 3.13 A screenshot illustrating the Control panel for a selected an item. In this example a red barrel is current selected.



a)



b)

Figure 3.14 Comparison between a) interface generated environment and b) real world environment.

4. Results and Discussion

The final system was evaluated to determine the systems stability and level of usability. These areas can then be subsequently reviewed in future work to improve the system. Stability was tested using test cases and usability was tested using a combination of user testing and heuristic evaluation.

4.1 Test Cases

Test cases were applied for both one and two robots for both standard and extreme cases, focusing on evaluating the performance of all the systems features. The test which was designed for this evaluation can be found in section 2.6.3. A full list of test case results can be found in Appendix C. For the single robot tests the system passed 50 out of 54 tests and for the double robot case 26 out of 31 tests, resulting in a combined score of 90%.

4.1.1 Stability

Through analysis of the test case numerical data it can be concluded that system performance can be considered stable for the purpose of a prototype system. However, this stability ranking cannot be extended to a higher level as very occasionally program crashing will occur. The cause of the crashing has not yet been directly identified. However, basic testing suggests the problem could be due to incorrect integration with rqt. More refinement will be needed to bring the system stability up to the level required for public release.

4.1.2 Discovered System Bugs

A number of system bugs where uncovered during the testing process with major bugs discussed below. For a complete list of all bugs see Appendix F.

An issue was identified when the robots began to leave the connection range of the network; system lag was experienced that slowed the full system. The reason for this is still not fully understood, however the current hypothesis is that the problem is due to a remote terminal window being kept open. The window is required to launch the robots and must remain running on the master computer. It is believed that as the connection strength decreases, system lag will occur due to this window. A solution to this would involve stating the robots without the need for a remote terminal window to be constantly open.

Another concern was the red barrel detection algorithm was found to be not robust under varying light sources. In environments of high light saturation no detection was registered. It was also identified that the robot motion was occasionally affecting the algorithm resulting in no detection. The user tests had originally planned to involve using the object detection. However, it was deemed that the detector would not be robust enough to produce consistent results as users will be conducting the tests in an environment with varying lighting conditions. Hence it was decided to remove the detection tasks from the user tests.

4.2 User Tests

Tests were conducted for a total of eight users, an outline of the user tests is explained in section 2.6.4.. Users came from a similar engineering background with varying levels of robotics experience. Three out of the eight users were considered to be experienced in robotics. The full test results for each user can be found in Appendix D.

4.2.1 Task Identification

The averaged results for the task identification tests are shown in Table 4.1. It should be noted this task was completed with the users having no prior knowledge of the system, only being told the system was based on a RTS game.

Task	**Time (s)**	**Wrong actions**	**Help needed**
Move the camera	<5	0.1	No
Zoom the camera in and out	<5	0	No
Select a robot	<5	0.1	No
Rotate a robot	<5	0	No
Find battery of robot 1	<5	0	No
Find the status of robot 3 camera	12.8	0.4	No
Find out what is wrong with robot 1	<5	0	No
Send robot a goal	<5	1	No
How would you stop a robot	5.2	0.4	No
Find out if an obstacle is in	41	6.6	Yes

	the robots path

Table 4.1 Averaged results from the Task Identification tests for a sample size of eight users, <5 refers to 'under five seconds'. Actions as defined as anything that can change the state the system.

The task identification results can be used to conclude that the user interface is providing an intuitive system for users to complete simple tasks. As a small sample size was used, with a limited number of tasks the results must be considered to be subjective. Because of this the conclusions made are to be considered as preliminary. To obtain more conclusive results a larger sample size must be used, with more detailed tasks of varying difficultly level and possibly conducted by an approved HCI tester (Sharp, et al., 2011).

4.2.2 Single Robot Test: Teleoperation vs Interface

The results from the single robot user tests were averaged and are shown in Table 4.2.

Type	Average Time (s)	Average Potential Crashes
Teleoperation	99.6	1.3
Interface	60.3	0

Table 4.2 Averaged results from the single robot control tests. A potential crash is defined as the test supervisor having to intervene to stop the user driving.

Results indicate that the interface provides an improvement of 35% in user times when driving to a destination, when compared to the teleoperation system. Users had on average 1.3 crashes when controlling the robot using teleoperation compared to zero crashes when using the interface. Users commented that they felt much more confident and in control when using the interface. Observations showed that users found it particularly difficult to navigate doorways and corners using teleoperation, this problem was non-existent using the interface. Figure 4.1 shows a comparison of times for both cases.

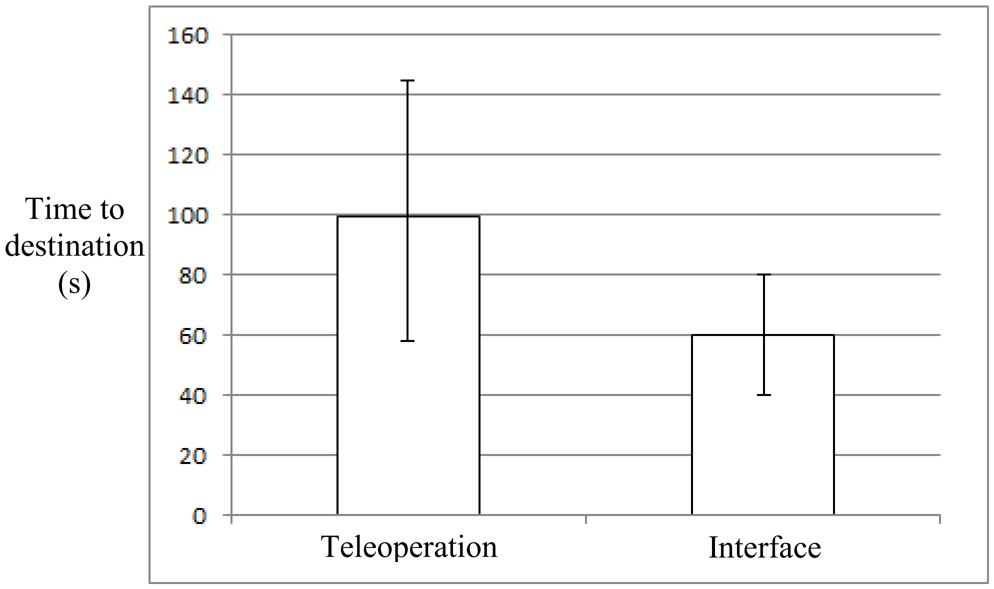


Figure 4.1 Mean time for users to drive to the destination point, a sample size of eight users. For teleoperation mean = 99.6s, slowest time = 145s, fastest time = 59s and for interface control mean = 60s, slowest time = 80s, fastest time = 40s. Error bars represent the fastest and slowest user times for the case.

4.2.3 Advanced Test

The advanced tests required users to map the third floor of the Electrical Engineering building using two robots. A full floor exploration was defined to include all the hall ways, the robotic laboratory and the staff room; all users where able to complete this task. A map generated by a user is shown in Figure 4.2. The analysed results can be seen in Table 4.3.

Type	**Result**
Average Map Time	277.7 s
Potential crashes	0.1
Fastest Time	210 s
Slowest Time	405 s

Table 4.4.3 Computed results from the user tests requiring users to map an area in the fastest time possible with two robots. A sample size of eight user was used.

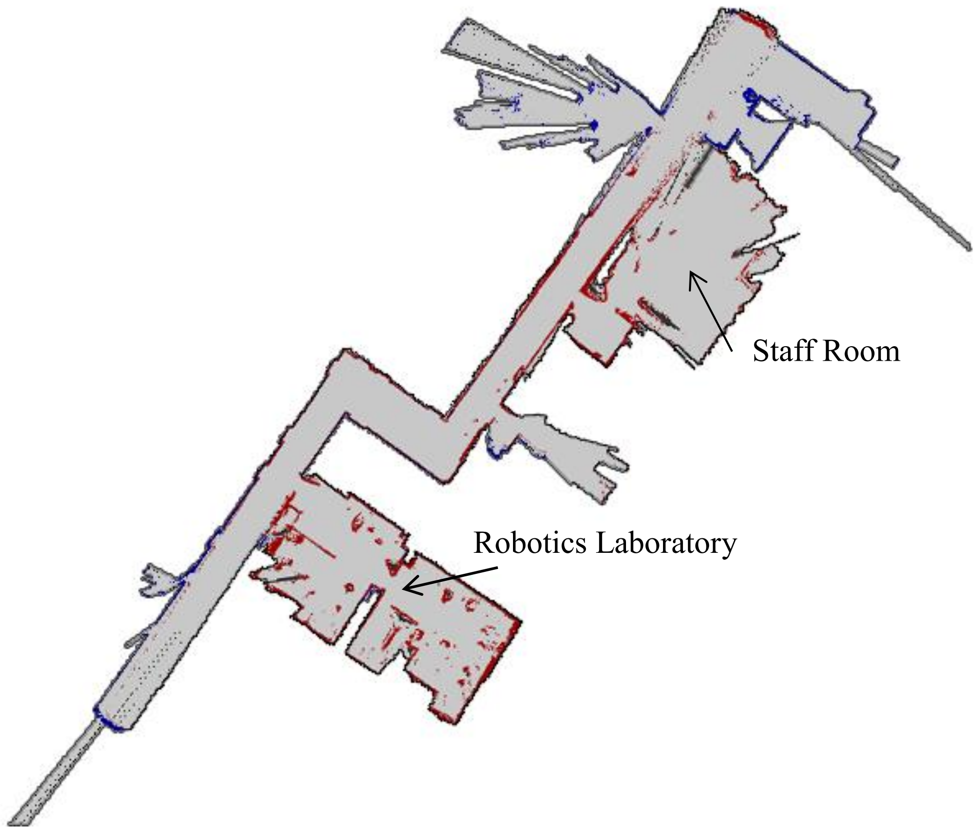


Figure 4.2 A user generated map of the third floor of the Electrical Engineering building using two robots. The robotics laboratory and staff room are labelled.

Users where divided into two groups, those with robotic experience and those without. A distinction was made by examining the user's academic and employment backgrounds. RTS experience cannot be used in the analysis as the method used to collect experience levels is subjective; requiring users to rank their abilities. A more advanced method required, such as getting users to play an RTS game resulting in a quantified score of performance. Figure 4.3 shows the difference in average time taken to map the floor between the different user types. The figure demonstrates that a user's performance is relativity constant and independent of the user's experience in the field of robotics. It should be noted that this test has a small sample size and contains possible outliers. Further testing will be required to produce conclusive.

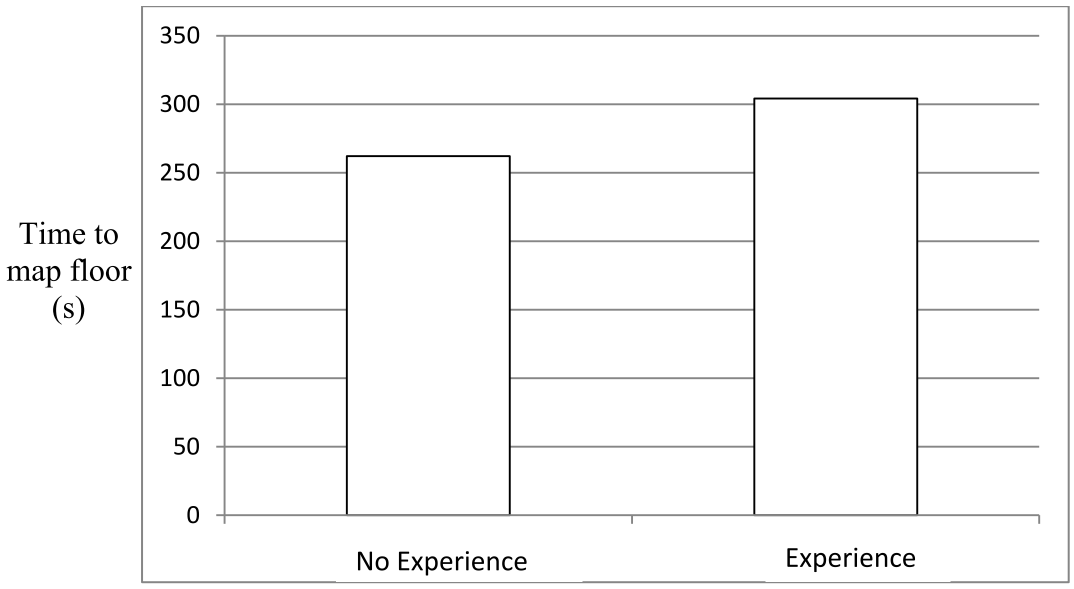


Figure 4.3 Mean time for user to map the third floor of the Electrical Engineering building. Comparison between uses with no robotics experience (mean = 262s) and users with robotic experience (mean = 304s).

4.2.4 Analysing User Comments and Observations

Analysing user feedback can be used to further infer the systems usability, details on each user's comments can be found in Appendix C. All users commented that they had more in control of the system and were more confident when using the interface as opposed to using teleoperation. Users commented that they liked the visual appearance of the interface and that information was appropriately organised and displayed.

4.3 <u>Heuristic Evaluation</u>

The data from user testing was later used to conduct a heuristic evaluation. The details explaining the evaluation can be found in section 2.6.5, the calculated heuristic scores are shown below in Table 4.4 below, with detailed explanation of the point allocations are located in Appendix E.

	Evaluation Criteria	Score
1	The overall navigation structure is clear and easy to understand	0
2	Minimalist design— No excess features or information	0
3	Visual elements (menus, colours, layout) are consistent across the system	0
4	Design follows same conventions as existing tools that serve the same or similar purpose	1
5	Interactions and behaviours are consistent across the system	0
6	Aesthetic integrity— Design has clear visual hierarchy, good alignment, good colours contrast, and readability	1
7	Each interface has a clear path for the user to take	0
8	Content is clear and concise	0
9	Labels and button text are clear and concise	0
10	Icons are unambiguous	1
11	UI elements provide visual feedback when manipulated (i.e. buttons depress when clicked)	0
12	Number of buttons/links is reasonable	0
13	System's visual characteristics imply how to interact with the interface (affordance)	1
14	Interface provides feedback to the user about the status of the system	2
15	System speaks the user's language	0
16	Users must confirm before error-prone actions are taken (i.e. deleting, quitting)	2
17	Users can tailor frequent actions to boost efficiency	2
18	Alerts and notifications are used only for the most important information	0
19	Error messages are in plain language, informative, and unambiguous	0
20	Error messages offer solutions	3
21	Users can easily access instructions, when appropriate	0
22	Documentation is easy to search, focused on the user's tasks, and not too dense	0
23	Layout could support translation to another language (i.e. Spacing is forgiving enough to accommodate longer words or characters	1
	Total Score	14

Table 4.4 Results from the heuristic evaluation. Detailed explanations of the point allocations are located in Appendix E.

4.3.1 Heuristic Evaluation Discussion

The interface scored 14 deducing points from the heuristic evaluation. This is equivalent to a score of 101/115 or 88%. This score can be used reinforce the belief that the interface is providing a highly usable experience for user. However, such a score can only be considered a preliminary estimation of the system. This is due to subjective nature of heuristic evaluation which is due to bias created from the author conducting the tests.

4.4 Important findings

4.4.1 Testing Limitations

As discussed in all test selections, the results obtained must be carefully assessed. The nature of many user interface evaluation methods is that they can produce inconclusive results for small sample sizes. Result accuracy can be improved by using professional experimenters, which can reduce the bias of tests or the use of a large and diverse user base. However, such methods were in excess of this project's allocated budget.

The best approach to obtain conclusive results for the developed system would involve a release to the robotic community. Once released, user feedback will be easily obtained, with the errors uncovered and subsequently repaired by system updates. The ROS package distribution system provides the ideal infrastructure to implement such a strategy.

4.4.2 Discovered Improvements

Possible system improvements can be identified through investigation of the obtained results. Stability analysis proved that an increase in robustness would be needed to produce a higher quality system. Further development of the communication system would also be needed, not only to increase the robustness of the current system, but to also increase the systems scalability.

Task identification results identified a number of improvements that will increase usability. Many users attempted to click either the hardware status icons or the robot state summary icons. Further refinement could improve this icon design to make it look less like a button. Some users did not understand the colour coding convention; this will have to be clearly illustrated in all manuals and help pages.

By analysing user comments and observations future improvements can be suggested. 100 % of users commented that the mouse drag system was hard to use at times, feeling that the motion was jerky and unresponsive. This component can be refined by using more advanced screen movement algorithms and will be addressed in future work. 63 % of users commented that the webcam controls could be improved, with some users suggesting that a hold function could be added. Such a feature would allow continuous camera movement as long as the button is pressed.

It was discovered through testing that additional feedback mechanisms will be required, with a specific area being the management of lag and delays. Commands sent to the robot will not happen instantly, requiring a small amount of time to be processed. 75% of users agreed that a connection strength meter for each robot will be useful, the simple 'yes or no' connection feature is not sufficient. It was identified that increased general system feedback was also needed. This would include mechanisms to let a user know their command has been sent and is being processed, as well as sounds and loading icons. When questioned about the addition of speakers and a microphone to the robots, 100% of users said this would be a great addition.

Other user suggested improvements included: a video recording feature, a health bar to represent robots connection strength, the introduction of hotkeys to allow a robot to be selected by pressing a number, having the camera track a robots position and the ability to send multiple goals to a robot.

4.4.3 System Limitations

The system was only tested with two robots, hence although the system was designed to accommodate for five robots, no claims can be made about the effectiveness of the system for more than two robots. The system was tested indoors and in a level environment. It is predicated that a 3D map display will be needed to allow for effective control in outdoor environments.

4.4.4 Comparison to the State of the art

Comparison to the state of the art is difficult as full access to another state of the art interface is required for a complete evaluation. Instead, to perform comparison primary features of each interface can be compared. It was determined through research that Team Michigan's interface (TMI) is currently deemed to be the most state of the art interface available. Hence this interface was used to compare against this projects developed interface (DI).

1) TMI requires two operators while DI needs only one.
2) TMI was been successfully tested using 11 robots compared to DI’s 2 robots.
3) DI focuses on providing a simple to use system and has been tested on users with no robotics experience or training, TMI has only been used by robotic experts.
4) TMI provides the option to show a three dimensional maps to help increase situational awareness, MRC does not.
5) DI provides a modular design that can be modified for different applications, TMI is specifically designed for explore and locate missions.

The above can be summarised as follows; the Team Michigan interface is a complex and powerful system, specifically tailored to the application of explore and locate missions. It however requires two highly trained operators and is built off private software, making it very inaccessible to outside researchers.

The developed interface is simpler in design and features when compared to TMI, but it is user friendly and can be operated by a single user. The system is built with a modular and ROS compatible design; meaning it can be easily integrated into other robotic systems running ROS.

5. Conclusions

The overall project objective was to develop an effective single-user, multi-robot user interface. The collection of project results can be used to create a preliminary evaluation of the system, which can be used to produce initial conclusions of the system.

Results from the test cases proved the system has adequate stability, noting that further refinement is still required. The single robot tests showed the system has improved performance over transitional teleoperation methods. Results from the task identification, user comments and heuristic evaluation prove the system is providing an intuitive and user friendly experience for the user. The advanced robot control tests demonstrate the system is useable for users inexperienced in in the field of robotics.

It can hence be concluded that the primary objective of creating a user-friendly, single user, multi-robot user interface, based upon the RTS paradigm has been developed. Demonstrating that for an indoor environment the system provides an implementation for a user of low robotics experience to effectively control two robots.

5.1 Contribution to the State of the Art

By developing the system as an rqt plugin and by using only ROS compatible components a system has been created that can be easily integrated into other ROS systems. Once released to the robotic community, the system will provide a state of the art and easy to run multi-robot interface that can be used by other researchers. By having access to ROS compatible product other researchers can use the system as the foundation for other interfaces, or to conduct further research in HRI. The project has also produced results for the use of the RTS paradigm in the design of a multi-robot interface. Preliminary results have begun to show that the RTS paradigm is an effective design constraint for the development of multi-robot user interfaces. However, more results are needed before conclusive claims can be made.

5.2 Multi-Robot User Interface Design Guidelines

A series of design guidelines can be recommend for the development of future multi-robot user interfaces. This has been developed from analysing the project results and incorporating the lessons learnt from the construction of the system.

5.2.1 Follow the RTS Paradigm

1) Use a layout based upon a central map display with a control panel placed below.
2) Implement a robot section method that continuously updates the control panel with data of the selected unit.
3) Look to tested RTS features when developing new interface features.

5.2.2 Enhance Situational Awareness

1) Use maps with an overlayed robot position to increase location awareness.
2) Overlay costmaps and provide a webcam feed to help increase surroundings awareness.
3) Provide users with a robot list which is visible at all times, to help increase mission awareness.

5.2.3 Lower users cognitive load

1) Provide a simple goal selection control system.
2) Implemented colour coding mechanisms.
3) Have the option to disable central map display features.
4) Use a clean and simple design, with well-defined borders, a simple colour palette and appropriate icons.

5.2.4 Adequate Status Information

1) Provide the user with easy to interpret robot status information.
2) Summarise and prioritise each robots state.
3) Use sounds and flashing animations to increase feedback.

6. Future Work and Recommendations

Using the current interface as a foundation a number of possible future work areas have been devised; with the primary focus on refining the system in preparation for release to the ROS community, and eventually for use in real world applications.

Further development focusing on increasing system robustness should ne the number one priority. Refining software on both the robots and master interface will ensure the system is stable in all situations. The development of a more advanced process manger will enable robots to effectively monitor all their nodes, taking informed actions to overcome errors, effectively increasing the robot robustness.

A more advanced communication system is needed. An asynchronous dynamic design that only sends data to the interface when requested should be implemented. For example a costmap should only be transmitted if the interface is actually displaying the map. Such a system will reduce the network traffic and will make the system more scalable, allowing for more robots to be used at one time. A mesh network could also be set up to allow the master to communicate with a robot by using the other robots as network nodes. This will greatly increase the systems communion range. Further user testing is required to correctly make conclusive conclusions which can later be published in support of the system upon its release. Using these new results and the results from the conducted tests further features refinement can be undertaken.

With a fully tested and refined multi-robot interface, work can turn to developing the system into a next generation system. Finishing the Map Builder port will allow for a state of the art large scale mapping solution that will greatly increase the maximum map size available. Converting the map module to a three dimensional map display will further increase the users situational awareness and will enable the system to work in more complex environments.

Finally application specific features can be added to tailor the system for a certain mission. Such as; developing object detection algorithms for search and rescue missions, multi-robot path navigation for mining applications and, durable hardware and software for military roles.

7. References

Banachowicz, K., 2011. *SICK ROS Drivers.* [Online]
Available at: http://www.ros.org/wiki/LMS1xx
[Accessed 2013].

Bevana, N., Kirakowski, J. & Maissel, J., 1991. *What is Usability?.* Stuttgart, Proceedings of the 4th International Conference on HCI.

Bruemmer, D. J. et al., 2005. *"Turn Off the Television!": Real-World Robotic Exploration Experiments with a Virtual 3-D Display.* s.l., Proc. HICSS.

Chen, J. Y., Haas, E. C., Pillalamarri, K. & Jacobson, C. N., 2006. *Human-Robot Interface: Issues in Operator Performance, Interface Design, and Technologies,* Aberdeen: U.S. Army Research Laboratory.

Crossman, J., Marinier, R. & Olson, E. B., 2012. *A Hands-Off, Multi-Robot Display for Communicating Situation Awareness to Operators.* Denver, Colorado, Proceedings of the International Conference on Collaboration Technologies and Systems.

Dixon, S. R. & Wickens, C. D., 2004. *Automation reliability in unmanned aerial vehicle flight control.* Mahwah, NJ: Lawrence Erlbaum, Proceedings of Human Performance, Situational awareness and Automation Conference.

Dixon, S. R., Wickens, C. D. & Chang, D., 2003. *Comparing quantitative model predictions to experimental data in multiple-UAV flight control.* Santa Monica, CA, Proceedings of the Human Factors and Ergonomics Society 47th Annual Meeting.

Drury, J. L., Keyes, B. & Yanco, H. A., 2007. *Lassoing HRI: analyzing situation awareness in map-centric and video-centric interfaces.* s.l., Proceedings of the Second ACM SIGCHI/SIGART Conference on Human-Robot Interaction.

Endsley, M. R., 1988. *Design and evaluation for situation awareness enhancement.* s.l., Proceedings of the Human Factors Society 32nd Annual Meeting.

Endsley, M. R. & Garland, D. J., 2000. *Situation Awareness: Analysis and Measurement.* s.l., LAWRENCE ERLBAUM ASSOCIATES.

Feil-Seifer, D., 2012. *p2os ROS drivers.* [Online]
Available at: http://www.ros.org/wiki/p2os
[Accessed 2013].

F. I., 2011. *Care-O-bot Environment.* [Online]
Available at:
http://ftp.isr.ist.utl.pt/pub/roswiki/cob_experimentation_days(2f)Tutorials(2f)Care(2d)O(2d)bot(20)Environment.html
[Accessed 2013].

Francois, S. & Gilbert, P., 2008. *Recent Advances in Multi Robot Systems, Chapter 10.* s.l.:s.n.

Gassend, B., 2012. *ROS Foreign Relay.* [Online]
Available at: http://www.ros.org/wiki/foreign_relay
[Accessed 2013].

Gerkey, B. P., Leibs, J. & Gassend, B., 2012. *ROS Hokyou laser scanner drivers.* [Online]
Available at: http://www.ros.org/wiki/hokuyo_node
[Accessed 2013].

Gomez, A. V., 2010. *Evolutionary Design of Human-Robot Interfaces for Teaming Humans and Mobile Robots in Exploration Missions,* Rome: University of Rome.

Goodrich, M. A. & Olsen, Jr, D. R., 2003. Seven Principles of Efficient Human Robot Interaction. *Systems, Man and Cybernetics,* Volume IV, pp. 3942-3948.

Hendrix, A., 2012. *ROS package audio_common.* [Online]
Available at: http://www.ros.org/wiki/audio_common
[Accessed 2013].

Jones, H. & Snyder, M., 2001. *Supervisory control of multiple robots based on a real-time strategy game interaction paradigm.* Tucson, AZ, Systems, Man, and Cybernetics, 2001 IEEE International Conference.

Kohlbrecher, S., 2012. *ROS SLAM example.* [Online]
Available at: http://answers.ros.org/question/35684/icp-scan-matching-failure/
[Accessed 2013].

Kohlbrecher, S. & Meyer, J., 2013. *ROS Hector Slam.* [Online]
Available at: http://www.ros.org/wiki/hector_slam
[Accessed 2013].

Lacaze, A., Murphy, K., Del Giorno, M. & Corley, K., 2012. Reconnaissance and Autonomy for Small Robots (RASR) team: MAGIC 2010 challenge. *Journal of Field Roboitcs,* 29(5), pp. 729-744.

Lazewatsky, D., 2011. *ROS cmvision.* [Online]
Available at: http://www.ros.org/wiki/cmvision
[Accessed 2013].

Marder-Eppstein, E., 2013. *ROS Navigation Stack.* [Online]
Available at: http://www.ros.org/wiki/navigation
[Accessed 2013].

Meeussen, W., 2011. *ROS multimaster.* [Online]
Available at: http://ros.org/wiki/multimaster
[Accessed 2013].

Mihelich, P. & Bowman, J., 2010. *ROS cv_bridge.* [Online]
Available at: http://www.ros.org/wiki/cv_bridge
[Accessed 2013].

Mihelich, P. & Bowman, J., 2013. *ROS Image Common.* [Online]
Available at: http://www.ros.org/wiki/image_common?distro=groovy
[Accessed 2013].

Nielsen, J., 1995. *10 Usability Heuristics for User Interface Design.* [Online]
Available at: http://www.nngroup.com/articles/ten-usability-heuristics/
[Accessed 2013].

Nielsen, J. & Molich, R., 1990. *Heuristic evaluation of user interfaces.* New York, SIGCHI Conference on Human Factors in Computing Systems.

Olson, E. et al., 2012. Progress Toward Multi-Robot Reconnaissance and the MAGIC 2010 Competition. *Journal of Field Robotics,* 29(5), pp. 762-792.

O. T., 2011. *Oxygen Iconset.* [Online]
Available at: http://www.iconarchive.com/show/oxygen-icons-by-oxygen-

icons.org.html
[Accessed 2013].

Qt Project, D., 2013. *Qt UI Framework.* [Online]
Available at: www.qt-project.org/
[Accessed 2013].

Qt, 2013. *QGLWidget Class Reference.* [Online]
Available at: http://qt-project.org/doc/qt-4.8/qglwidget.html
[Accessed 2013].

Reid, R. & Braunl, T., 2011. *Large-scale Multi-robot Mapping in Magic 2010.* Qingdao, IEEE, pp. 239-244.

Rule, A. & Forlizzi, J., 2012. Designing Interfaces for Multi-User, Multi-Robot Systems. *Human-Robot Interaction: Proceedings of the seventh annual ACM/IEEE international conference,* pp. 97-104.

Schipani, S. P., 2003. *An evaluation of operator workload during partially-autonomous vehicle operations.* s.l., Proceedings of PerMIS 2003.

Sharp, H., Rogers, Y. & Preece, J., 2011. *Interaction Design: Beyond Human-computer Interaction.* 3rd ed. s.l.:John Wiley & Sons.

Spaepcke, 2013. *Rviz.* [Online]
Available at: http://www.ros.org/wiki/rviz
[Accessed 27 05 2013].

Steffi, 2013. *Usability Heuristics.* [Online]
Available at: http://wiki.osrfoundation.org/Usability/Heuristics
[Accessed 2013].

Steinfeld, A. et al., 2006. *Common metrics for human-robot interaction.* s.l., Proceeding of the 1st ACM SIGCHI/SIGART conference on Humanrobot interaction.

Stonier, D., 2013. *ROS Multimaster Special Interest Group.* [Online]
Available at: http://www.ros.org/wiki/sig/Multimaster
[Accessed 2013].

Thomas, D., 2013. *ROS PyStyle Guide.* [Online]
Available at: http://www.ros.org/wiki/PyStyleGuide
[Accessed 2013].

Thomas, D., Scholz, D. & Blasdel , A., 2013. *RQT.* [Online]
Available at: http://www.ros.org/wiki/rqt
[Accessed 2013].

Trouvain, B. & Wolf, H. L., 2003. *Design and Evaluation of a Multi-Robot Control Interface.* s.l., NATO.

Wang, J. et al., 2008. *Experiments in Coordination Demand for MultiRobot Systems.* s.l., Proceedings of 2008 IEEE International Conference on Distributed Human-Machine Systems.

W. G., 2012. *ROS.* [Online]
Available at: http://www.ros.org/wiki/
[Accessed 2012].

8. Appendices

Appendix A – Graphical Interface Evolution

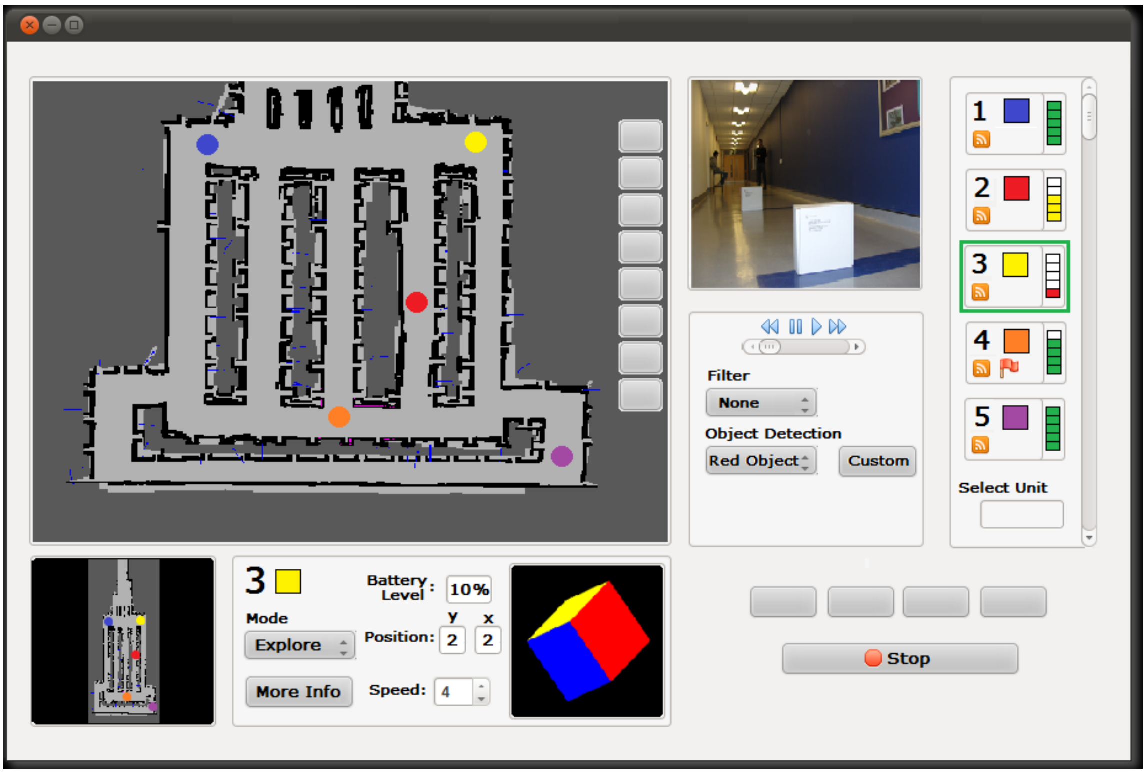


Figure A1 Graphical user interface version 1, a mock up desinged in photoshop.

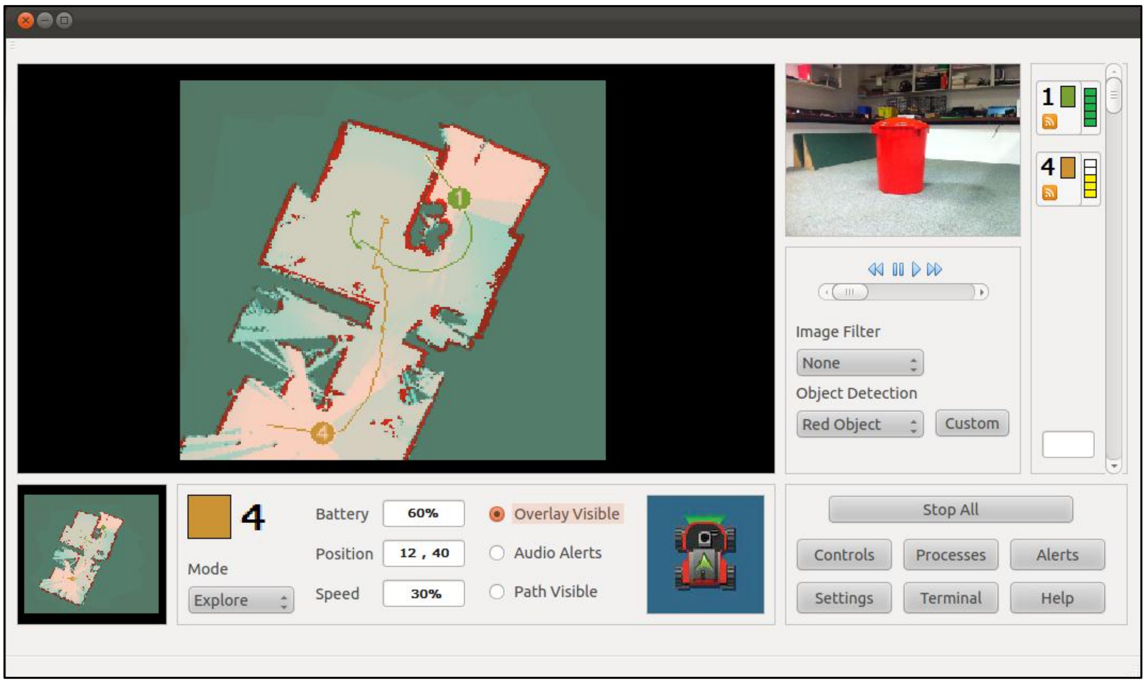


Figure A2 Graphical user interface version 2, layout designed in Qt. Other features not functional, images added with Photoshop.

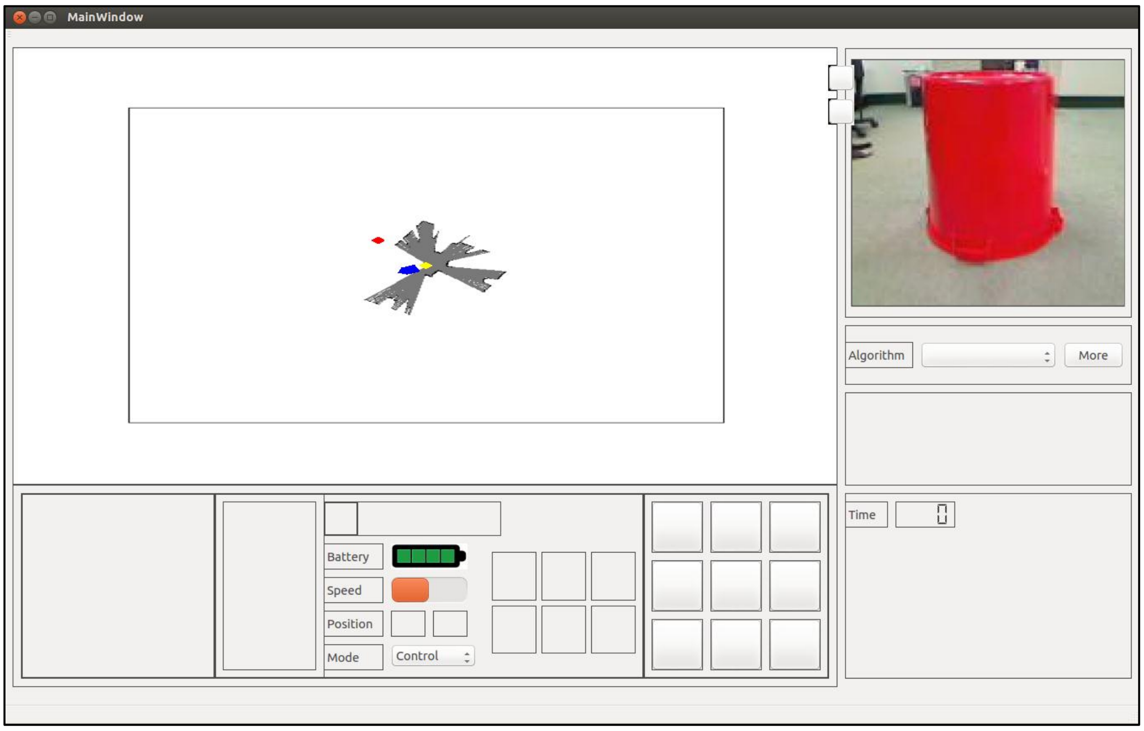


Figure A3 Graphical interface version 3, functional; map module, video feed, robot selection and goal commands. Basic control panel functions where implemented.

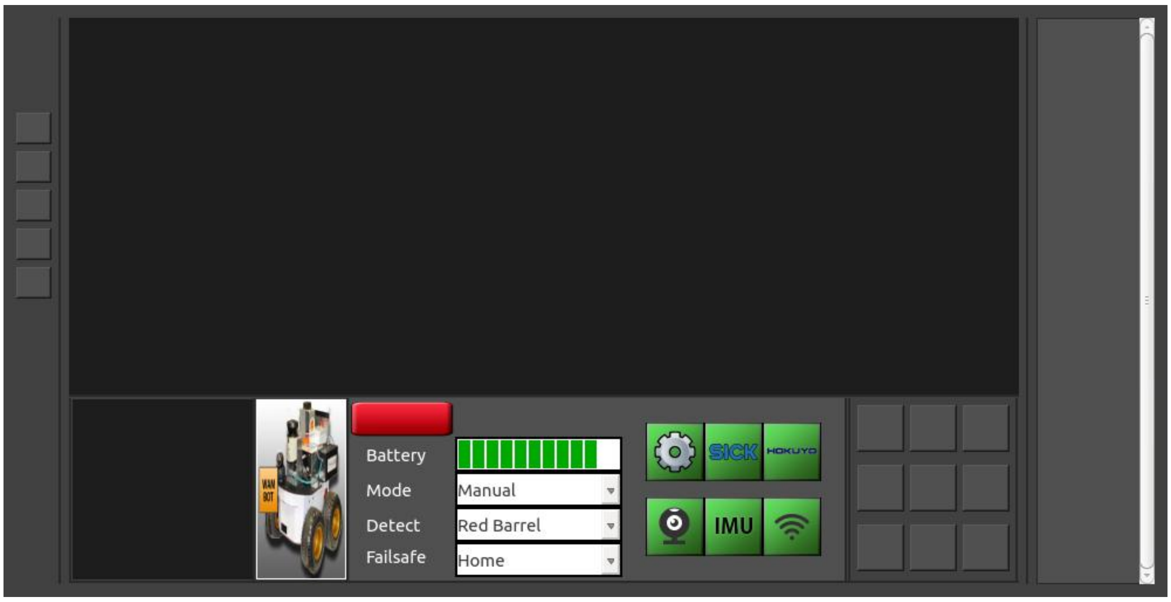
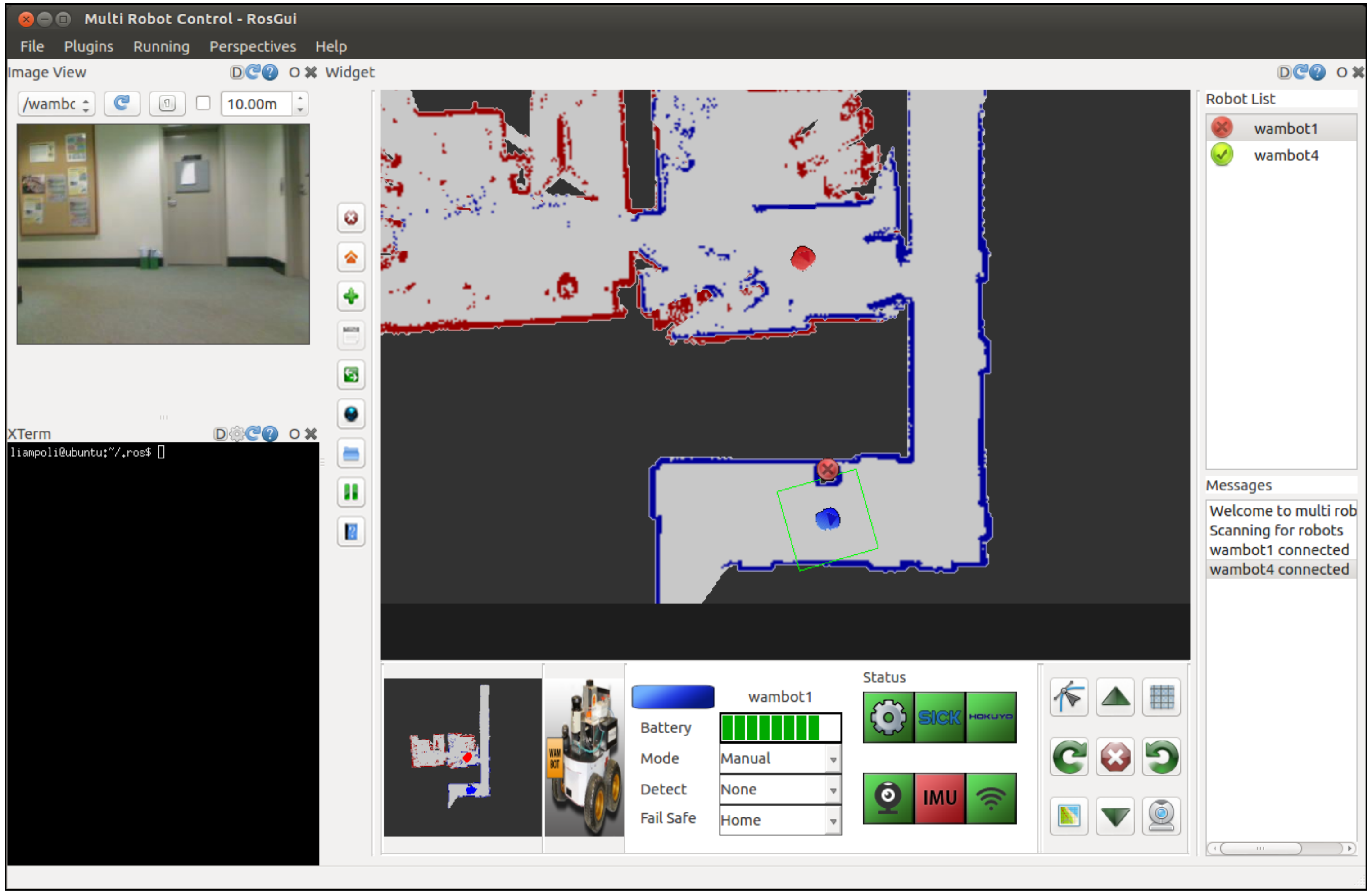


Figure A4 Graphical interface version 4, non-functional purely a layout and visual design. Colour palette had to be abandoned as to be compatible with the rqt framework.



```
liampoli@ubuntu:~/.ros$
```

Welcome to multi rob
Scanning for robots
wambot1 connected
wambot4 connected

wambot1

Battery	
Mode	Manual
Detect	None
Fail Safe	Home

Figure A5 Graphical interface version 5, a fully working system embedded within the rqt framework.

Appendix B - Custom Messages

base_to_robot

```
int64 type
string message
```

robot_to_base

```
Header header
string owner
int8[] status
string message
```

ObjectOfIntrest

```
int8 type
string name
string finder
time found_time
sensor_msgs/Compressed image
```

Type	**Description**
1	Change detection algorithm
2	Send a global goal
3	Send a local goal
4	A drive command
5	Change robots mode
6	Change robot fail safe
7	Connection check

Table B1 Key table used by the base_to_robot message.

Type	**Type Name**	**Sub Type**	**Description**
5	Change robots mode	0	Manual Control
5	Change robots mode	1	Explore
1	Change detection algorithm	0	Nothing
1	Change detection algorithm	1	Red Barrels
1	Change detection algorithm	2	Human faces
6	Change robot fail safe	0	Home
6	Change robot fail safe	1	Explore
6	Change robot fail safe	2	Stop

Table B1 Sub key used to define which selection with in a state change command. For example is robot mode is to be changed, 0 supplied to the message field will command the robot to enter manual control.

Appendix C - Test Case Results

C1 One Robot

	Test Case	Result	Comments
1	Turn on wambot4	PASS	-
2	Start-up interface	PASS	-
3	Zoom map	PASS	-
4	Move map	PASS	Could do with refinement
5	Toggle template	PASS	-
6	Move template	PASS	-
7	Confirm robot state is correct	PASS	-
8	Toogle map control	PASS	-
9	Zoom mini map	PASS	-
10	Select robot, unselect robot	PASS	-
11	Confirm robot battery	PASS	-
12	Confirm robot battery has three colours	PASS	-
13	Confirm robot state	PASS	-
14	Fail primary laser – check status icon	PASS	-
15	Fail secondary laser– check status icon	PASS	-
16	Fail camera– check status icon	PASS	-
17	Fail IMU – check status icon	FAIL	Not implemented
18	Lose connection – check icon	PASS	-
19	Check base warning icon	PASS	-
20	Check secondary laser warning icon	PASS	-
21	Send robot goal	PASS	-
22	Toggle costmap	PASS	-
23	Toggle path	PASS	-
24	Turn map black	PASS	-
25	Turn map off	PASS	-
26	Stop robot	PASS	-
27	Rotate left and right	PASS	Could increase rotation amount
28	Move forward and back	PASS	More feedback as to what the robot is doing could be helpful.
29	Toggle camera controls	PASS	-
30	Move camera	PASS	-
31	Add item	PASS	-
32	Move item	PASS	-
33	Select item, then select robot	PASS	-
34	Remove item	PASS	-

35	Force robot to state summary 1 - no connection	PASS	-
36	Force robot to state summary 2 – low power	PASS	-
37	Force robot to state summary 3 - error	PASS	-
38	Force robot to state summary 4 - Warning	PASS	-
39	Force robot to state summary 5 - OK	PASS	-
40	Confirm detect state change	PASS	-
41	Confirm red barrel detection – low light	FAIL	-
42	Confirm red barrel detection – standard light	PASS	-
43	Confirm red barrel detection – highly saturated	FAIL	-
44	Confirm item displayed on map, confirm item shown	PASS	-
45	Move found item	PASS	-
46	Delete item	PASS	-
47	Confirm face detection	PASS	-
48	Confirm failsafe state change	PASS	-
49	Confirm home failsafe	PASS	-
50	Confirm none failsafe	PASS	-
51	Confirm stop failsafe	PASS	-
52	Explore area with interface	PASS	-
53	Explore using controller	PASS	-
54	Move robot out of range	FAIL	System experienced lag and did not update the interface with the lost connection.

Table C1 Results from the single robot test cases

Pass rate = 50 /54 = 92.6%

C2 Two Robots

	Test Case	**Result**	**Comments**
1	Turn on wambot1 (robot 1)	PASS	-
2	Turn on wambot4 (robot 2)	PASS	-
3	Confirm robot 1 state is correct	PASS	-
4	Confirm robot 2 state is correct	PASS	-
5	Select robot 1	PASS	-
6	Select robot 2	PASS	-

7	Send robot 1 a goal	PASS	-
8	Send robot 2 a goal	PASS	-
9	Stop robot 1	PASS	-
10	Stop robot 2	PASS	-
11	Stop all robots	PASS	-
12	Toggle robot 1 costmap	PASS	-
13	Toggle robot 2 costmap	PASS	-
14	Toggle robot 1 path	PASS	-
15	Toggle robot 2 path	PASS	-
16	Move robot 1 camera	PASS	-
17	Move robot 2 camera	PASS	-
18	Test switching video feed	FAIL	-
19	Test joy stick controller for both	FAIL	-
20	Explore with robot 1	PASS	-
21	Explore with robot 2	PASS	-
22	Drive robot 1 into a low connection range	FAIL	-
23	Drive robot 2 into a low connection range	FAIL	-
24	Return robot 1 home	PASS	-
25	Return robot 2 home	PASS	-
26	Select goal in unknown space	PASS	-
27	Return camera home	PASS	-
28	Press help button	PASS	-
29	Pause map update and confirm it is correct	PASS	-
30	Stop all robots	PASS	-
31	Send both robots to the same goal and confirm no collision occurs	FAIL	-

Table C2 Results from the test case with two robots.

Pass rate = 26/31 = 83.8%

Appendix D – User Test Results

D1 User 1

Name	Dagogo Altraide
Age	22
Qualifications	Mechanical Engineer
Robot Experience	None – 0
RTS experience	Limited - 2

Task Identification

Task	Time	Wrong actions	Help needed	Comments
Move the map view	U5	0	No	-
Zoom the map view	U5	0	No	-
Select a robot	U5	0	No	-
Rotate a robot	13	0	No	Took a bit of time to find the button.
Find battery of robot 1	U5	0	No	-
Find the camera status of robot 2	20	1	No	Had to be told it was in the control panel
Find out what is wrong with robot 1	U5	0	No	Previous question influenced?
Send a goal to robot 1	U5	3	No	Pressed the wrong button
Stop a robot	U5	0	No	-
Find out if an obstacle is in the robots path	NA	9	Yes	Over 60 seconds

Teleportation vs. Interface

Type	Time	Potential Crashes
Teleoperation	124 s	2
Interface	68 s	0

Advanced Test

Type	**Result**
Map Time	302 s
Potential crashes	0

User Comments and Observations

The user stated that he was very inexperienced with real time strategy games. In the task identification test the user required help completing some tasks.

In the single robot test the user found it difficult to drive the robot though the initial door way, driving very slowly and carefully. Even with his caution two potential crashes occurred in his attempt to clear the door way. The user commented that he would like to see between camera controls, "The camera control was sub-par". He suggested adding a function where a user could hold the button to allow a continuous camera movement.

For the advanced test the user showed relatively good control of the system. Although his exploration method was sub optimal. The user commented about the map drag function saying it needed more work, "The map drag is very jerky and hard to control". During this user test the hector mapping algorithm on the robot produced a slightly messy map. The user commented that this was confusing.

D2 User 2

Name	Prawi Woods
Age	22
Qualifications	Civil Engineering student
Robot Experience	None - 0
RTS experience	Very experienced real time strategy game player, expert StarCraft player - 9

Task Identification

Task	Time	Wrong actions	Help needed	Comments
Move the map view	U5	1	No	First attempt didn't hold button
Zoom the map view	U5	0	No	-
Select a robot	U5	0	No	-
Rotate a robot	U5	0	No	-
Find battery of robot 1	U5	0	No	-
Find the camera status of robot 2	10	1	No	Did not understand the colour coding.
Find out what is wrong with robot 1	U5	0	No	-
Send a goal to robot 1	U5	0	No	Pressed the wrong button
Stop a robot	U5	0	No	-
Find out if an obstacle is in the robots path	30 s	3	Yes	Was told to try press buttons

Teleportation vs. Interface

Type	Time	Potential Crashes
Teleoperation	**80**	**1**
Interface	**52**	**0**

Advanced Test

Type	**Result**
Map Time	248
Potential crashes	0

User Comments and Observations

In the task identification tests the user seemed very confident using the system, being able to complete most tasks very quickly. The user had to be told the colour coding of the status icon, but was able to identify that the icon represented the camera status. Did not know what a costmap was, but was able to identify that it was showing a real time obstacle map after clicking it.

In the single robot test the user found it hard to drive though doors, moving slowly when taking corners, "I found it very hard to control the robot especially around corners," and "I wasn't sure which direction the camera was pointing with respect to the robots orientation". The user commented that he really enjoyed used the system and that it reminded him of playing an RTS game, "I really like this it, it really does remind me of playing Starcraft."

For the advanced test the users experience in RTS gaming was apparent, with the user making fast and decisive decisions. The user said he enjoyed using the system and likes the idea of basing the system off an RTS. When questioned about what the user thought of the overall system his replay was "The system works, but it still needs refinement". User suggestions included; Keep a unit selected when moving the map, be able to set multiple goals, highlight the drive command buttons as the robot is driving. After 20 – 30 minutes of controlling the robots, the user claimed he was comfortable using the system.

D3 User 3

Name	Phillip Whyte
Age	22
Qualifications	Mechanical Engineer with Finance Degree
Robot Experience	None – 0
RTS experience	Very Limited - 2

Task Identification

Task	Time	Wrong actions	Help needed	Comments
Move the map view	U5	0	No	-
Zoom the map view	U5	0	No	-
Select a robot	U5	0	No	-
Rotate a robot	U5	0	No	-
Find battery of robot 1	6 s	0	No	-
Find the camera status of robot 2	U5	1	No	Understood the colour coding for green equals OK
Find out what is wrong with robot 1	U5	0	No	Figured immediately the IMU error
Send a goal to robot 1	U5	2	No	Took a few attempts
Stop a robot	20	3	No	Tried to click the error icon button
Find out if an obstacle is in the robots path	52 s	8	Yes	Through trail and error the user figured out he needed to pressed the costmap button.

Teleportation vs. Interface

Type	Time	Potential Crashes
Teleoperation	75 s	3
Interface	41 s	0

Advanced Test

Type	**Result**
Map Time	210 s
Potential crashes	1

User Comments and Observations

For the task identification test the user performed well given his limited experience with RTS video games. The user correctly identified the colour coding of green equals OK and red for error. The user was confused by the error icon above the robot, "I thought the error icon was a stop button". User also thought the status icons where buttons.

User produced fast times for both single robot cases, although an aggressive approach was taken in the teleportation case, which can be seen in the three potential crashes. It should be noted that in the interface case the user used both a combination of setting goals and using the joy stick controller. The controller was used to get around convers faster and make small adjustments to the path. This proved to be a very good combination and resulted in a very fast time. The user commented that he could found interface control much easier, " because I could click on the goals and see the map, this made it much easier to go thought doors. Because I could see the width of the door and then compare it to the size of the robot I was confident the robot could clear it and thus could go faster". This comment is illustrating that the user is experiencing increased situational awareness.

The user commented that he found the first test more exciting, "I like the immersive nature of using only a video feed and it felt like I was going faster". However the user commented that he found the second test more "satisfying". User suggestions on how to improve the interface include; more feedback, change the robot icon, "I can't see the arrow very well", a heading on the video feed to both show the robot name and if a feed is streaming, a video record function, wants speakers so can interact with surrounding people.

User was impressed with the system and liked how the map would update as the robot explored. User didn't like the lag that occurred as the robots got further away from the base, suggested a connection icon to show the connection strength of a robot.

D4 User 4

Name	Bradley Pattrick
Age	23
Qualifications	Chemical Engineer and Chemist
Robot Experience	None - 0
RTS experience	Very Experienced – 7

Task Identification

Task	Time	Wrong actions	Help needed	Comments
Move the map view	U5	0	No	-
Zoom the map view	U5	0	No	-
Select a robot	U5	1	No	Clicked the error icon.
Rotate a robot	U5	0	No	-
Find battery of robot 1	10 s	0	No	Thought the robot colour icon was the charge.
Find the camera status of robot 2	43 s	0	1	User to a long time to figure out the status icons represented hardware state. Did not understand colour coding.
Find out what is wrong with robot 1	10 s	0	No	Influenced by help in previous section.
Send a goal to robot 1	U5	1	No	Clicked wrong mouse button
Stop a robot	U5	0	No	-
Find out if an obstacle is in the robots path	NA	10	-	User could not figure out in fewer than 10 attempts.

Teleportation vs. Interface

Type	Time	Potential Crashes
Teleoperation	109 s	0
Interface	78 s	0

Advanced Test

Type	Result
Map Time	315 s
Potential crashes	0

User Comments and Observations

For the task identification test user could not figure out the status colour coding, and commented that the colour of green for OK and red for error is not easy to figure out. However once the understood he said "the convention makes sense".

For the single robot tasks the user drove very slowly thought the door. Commenting "the joy stick control was more fun and liked how it went faster". However he also said "he felt less safe and confident with the joy stick" and proposed an idea of side mounted cameras.

For the advanced test the user showed good multi-tasking stills being able to effectively and quickly control the two robots. In the users test the map produced was particularly messy, the user responded well to this error and was able to complete the task. Said the map drag needed more refinement and was hard to use, said the scroll was too fast, didn't like hoe the hector map got messy. Said he likes the goal setting system and that it was easy to use. Said the responsiveness of the system was adequate. Noted that he did not use the mini map and instead simply zoomed out on the main map.

D5 User 5

Name	Remi Kent
Age	22
Qualifications	Software Engineer
Robot Experience	Highly experienced with ROS. Has worked on the Wambots and a Nao - 7
RTS experience	High level of experience with video games. - 7

Task Identification

Task	Time	Wrong actions	Help needed	Comments
Move the map view	U5	0	No	-
Zoom the map view	U5	0	No	-
Select a robot	U5	0	No	-
Rotate a robot	U5	0	No	-
Find battery of robot 1	U5	0	No	-
Find the camera status of robot 2	7 s	0	1	Told to look in display panel
Find out what is wrong with robot 1	U5	0	No	-
Send a goal to robot 1	U5	1	No	-
Stop a robot	U5	0	No	-
Find out if an obstacle is in the robots path	NA	10	-	User could not figure out in fewer than 10 attempts.

Teleportation vs. Interface

Type	Time	Potential Crashes
Teleoperation	145 s	2
Interface	80 s	0

Advanced Test

Type	Result
Map Time	247 s
Potential crashes	0

User Comments and Observations

User produced very quick identification of all tasks. The user did not manger to identify the costmap could be used to pick up obstacles.

When driving the single robot using teleoperation the user found it quite hard to drive thought the door taking a long time. The user had two potential crashes when attempting to drive thought the door. The user commented how he went the map was moved the unit was unselected. Commented "the interface was must easier than driving with the controller".

User suggested displaying the robots name or number above the map icon. User suggested a health bar above the robot icon that would represent the connection strength of the robot to the base. When asked what the user thought of the overall system he replied "I really like the system and found it very easy to use.".

D6 User 6

Name	Enda Mccauley
Age	21
Qualifications	Mechatronics Engineering and Computer Science student
Robot Experience	Highly experienced with ROS. Is working on a project using the Wambots. - 7
RTS experience	Reasonable - 5

Task Identification

Task	Time	Wrong actions	Help needed	Comments
Move the map view	U5	0	No	-
Zoom the map view	U5	0	No	-
Select a robot	U5	0	No	-
Rotate a robot	6	0	No	-
Find battery of robot 1	U5	0	No	-
Find the camera status of robot 2	7 s	0	1	Correctly identified green as OK.
Find out what is wrong with robot 1	U5	0	No	-
Send a goal to robot 1	U5	1	No	Clicked wrong mouse button to send goal
Stop a robot	U5	0	No	-
Find out if an obstacle is in the robots path	30	10	-	Was told to try press buttons.

Teleportation vs. Interface

Type	Time	Potential Crashes
Teleoperation	140 s	2
Interface	75 s	0

Advanced Test

Type	**Result**
Map Time	405 s
Potential crashes	0

User Comments and Observations

User performed very well in the task identification test this test being able to complete most tasks very quickly. User failed to identify the cost map was a way to determine real time obstacles.

For the single robot control test the user was very cautious when going thought doorways, drove constantly too close to the walls and commented “It was very hard to dive the robot through a door way”.

User performed advanced test slowly although appeared to have good control and understanding of the system. The mapping in this test produced a messy overlay. The user overcame this easily saying “the map colour coding and the fact that the selected map is displayed on the highset layer made it easy enough to comprehend the environment”. At one stage the user hid one of the robot maps so that he could better control the other robot.

User suggested adding hot keys so that robots could be selected by pressing a number. He also suggested that when the number was pressed twice the camera would centre on the robot. The user commented that really liked the system and that it remained him a lot of playing an RTS.

D7 User 7

Name	Calum Meiklejoh
Age	23
Qualifications	Robotics Engineer
Robot Experience	Has previously worked on the Wambots. - 9
RTS experience	Average – 5

Task Identification

Task	Time	Wrong actions	Help needed	Comments
Move the map view	U5	0	No	-
Zoom the map view	U5	0	No	-
Select a robot	U5	0	No	-
Rotate a robot	U5	0	No	-
Find battery of robot 1	U5	0	No	-
Find the camera status of robot 2	U5	0	No	-
Find out what is wrong with robot 1	U5	0	No	-
Send a goal to robot 1	U5	0	No	-
Stop a robot	U5	0	No	-
Find out if an obstacle is in the robots path	15	10	-	-

.

Teleportation vs. Interface

Type	Time	Potential Crashes
Teleoperation	59s	0
Interface	49s	0

Advanced Test

Type	Result
Map Time	260s
Potential crashes	0

User Comments and Observations

This user way identified to have a very high level of robotic experience, having previously worked on a project with the Wambots. The user performed very well completing all tasks quickly, with no mistakes and no assistance.

The user commented that he was experienced controlling the robot via a video feed, this showed in the users time. User said “I would like to be able to play sounds as I am exploring”. When asked about possible improvements the user suggested; improving the map drag, improve the navigation path finding, longer lasting batteries and a improved Wi-Fi network.

The user commented “this is a very easy to use system and I really like it”. He also made a comparison to Rviz saying “It is much easier to control compared to Rviz”.

D8 User 8

Name	Calum Webb
Age	22
Qualifications	Mechanical Engineer and Computer Scientist
Robot Experience	Low. – 2
RTS experience	Very High - 8

Task Identification

Task	Time	Wrong actions	Help needed	Comments
Move the map view	U5	0	No	-
Zoom the map view	U5	0	No	-
Select a robot	U5	0	No	-
Rotate a robot	U5	0	No	-
Find battery of robot 1	U5	0	No	-
Find the camera status of robot 2	10 s	0	1	Tried to click the status icon.
Find out what is wrong with robot 1	U5	0	No	-
Send a goal to robot 1	U5	0	No	-
Stop a robot	U5	0	No	-
Find out if an obstacle is in the robots path	25	3	-	Was able to quickly figure out what to do by pressing buttons.

.

Teleportation vs. Interface

Type	Time	Potential Crashes
Teleoperation	65 s	1
Interface	40 s	0

Advanced Test

Type	Result
Map Time	235 s
Potential crashes	0

User Comments and Observations

User completed the task identification task quickly and will few mistakes. When driving the robot with teleoperation the user was appeared confident although the path the robot took was slightly sporadic. The user commented "I felt much more in control using the interface" also stating "The video feed was more fun to use".

The user suggested some possible improvements; keep a unit selected when the map is moved, refine the map drag, have an option that makes the camera follow a robots position, be able to set multiple waypoints, hotkeys for robot section the addition of speakers and a microphone. The user conducted by saying "Overall the system was really good and easy to use, however it needs more refinement before it can be published".

Appendix E – Heuristic Evaluation Reasoning

The following provides reasoning to the scores allocated in the heuristic evaluation.

1) The overall navigation structure is clear and easy to understand

All users were able to complete the task identification challenges in appropriate times.

Score of **0** assigned.

2) Minimalist design— No excess features or information

Users commented on the clean and simple layout

Score of **0** assigned.

3) Visual elements (menus, colours, layout) are consistent across the system

Two major sets of icons where used, all with a simple colour palette .This increases icon design consistency. The status colour coding was keep constant, red for error, yellow for warning and green for OK.

Score of **0** assigned.

4) Design follows same conventions as existing tools that serve the same or similar purpose

As new icons and conventions are introduced the system cannot be said to fully be fully consistent with the current rqt interface conventions. The use of hints makes this a problem a purely cosmetic flaw.

Score of **1** assigned.

5) Interactions and behaviours are consistent across the system

The control system is identical for all unit types. Select is obtained from clicking the units map icon and commands to the unit can be issued from the command pad.

Score of **0** assigned.

6) Aesthetic integrity— Design has clear visual hierarchy, good alignment, good colours contrast, and readability (good font size & spacing)

Design is not fully optimised and can benefit from further refinement. Improvements would be purely cosmetic changes and would not significantly affect the usability of the system.

Score of **1** assigned.

7) Each interface has a clear path for the user to take
   User testing proved that for simple tasks user where able to clearly identify the path to take to complete an action. Hints are available to help a user determine the correct path to a solution.

Score of **0** assigned.

8) Content is clear and concise

System visual dsing is clean and simple. Icons and colour are used to simplify data to reduce the cognitive laoad on the user.

Score of **0** assigned.

9) Labels and button text are clear and concise- 0

All texted used is short and to the point. Hints provide the an appropriate level of text.

Score of **0** assigned.

10) Icons are unambiguous

Task identification showed that users did not always correctly identify icons. Further refinement of icon design would be beneficial.

Score of **1** assigned.

11) UI elements provide visual feedback when manipulated (i.e. buttons depress when clicked)

Interface provides adequate forms of feedback for tasks, buttons are depressed when pressed, sounds are produced to correspond with actions and when a robot goal is given an icon is placed on the map.

Score of **0** assigned.

12) Number of buttons/links is reasonable

Users commented on the clean and simple design of the interface. The number of buttons was deemed to be reasonable.

Score of **0** assigned.

13) System's visual characteristics imply how to interact with the interface (affordance)

Most of the interface proved to be intuitive, users recognised buttons as buttons, however many users thought the status icons where buttons and proceeded to click these. Further refinement would benefit the system.

Score of **1** assigned.

14) Interface provides feedback to the user about the status of the system

State summaries are given and messages are displayed to the user to indicate a change in the system. More refinement would be beneficial, adding flashing animations and sounds to indicate a state change would increase usability.

Score of **2** assigned.

15) System speaks the user's language (appropriately technical for the audience

User tests showed that even inexperienced users where able to complete complex tasks with the system.

Score of **0** assigned.

16) Users must confirm before error-prone actions are taken (i.e. deleting, quitting)

The deleting of false item identification does not require user confirmation. The fix to this will only require a minor software modification.

Score of **2** assigned.

17) Users can tailor frequent actions to boost efficiency, where applicable

No mechanism to increase efficiency of frequent actions was included. One solution to this would be to add hotkeys, such an implantation would be a minor change thus a score of 2.

Score of **2** assigned.

18) Alerts and notifications are used only for the most important information

Alerts are prioritised so that the user only sees the most important information.

Score of **0** assigned.

19) Error messages are in plain language, informative, and unambiguous

All error messages supplied are intuitive and informative.

Score of **0** assigned.

20) Error messages offer solutions

Error messages do not offer solutions. As solutions too many errors are quite complex a feature to work out error solutions would require considerable work.

Score of **3** assigned.

21) Users can easily access instructions, when appropriate

Users can easily access the help page by clicking on the help button located on the system control panel. Hints provide advice if users are unsure of a button or icons function.

Score of **0** assigned.

22) Documentation is easy to search, focused on the user’s tasks, and not too dense - 0

Documentation can be accessed thought the help button. Instructions are easy to follow and offer pictorial descriptions.

Score of **0** assigned.

23) Layout could support translation to another language (i.e. Spacing is forgiving enough to accommodate longer words or characters

The interface was designed making all Qt features scalable. This means items can be easily resized while keeping the interface proportions. This system was not tested with other languages and thus will still benefit from refinement.

Score of **1** assigned.

Appendix F – Bug Tracker

The following Appendix contains a list of discovered bugs that where uncovered during the testing phase.

F1 Major Bugs

1) Red barrel detector is not robust under varying lighting conditions.
2) As the connection between the robot and master weakens the system develops lag.
3) A crash can occur when the plugin is moved within the rqt frame.

F2 Minor Bugs

1) Mag drag is at times hard to control.
2) Video selection method is not current working.
3) The rotation command for the robots will some time not cause an incremental movement but a continuous movement.
4) Sometimes paths will be calculated through unknown space, causing un-expected results.
5) The communication hardware icon sometimes does not update.
6) The scan feature has problems.
7) The toggle highlight sometimes will not turn off.