2012

Colin Smith
Matthew Babiszewski
Jason Kerslake

[ROS GUI PROJECT CHARTER]

[This Page has been intentionally left blank]

Table of Contents

1. Executive summary 4
2. Purpose of the Project Initiation Document 5
3. Competitive Intelligence Support 6
   - 3.1 Purpose of the project 6
   - 3.2 Benefits 7
   - 3.3 Limitations 8
   - 3.4 Critical Success Factors 9
4. Scope 10
   - 4.1 In Scope 10
   - 4.2 Out of Scope 11
5. Project Plan 12
   - 5.1 Project Approach 12
   - 5.2 Project Schedule & Milestones 13
   - 5.3 Deliverables 15
   - 5.4 Communication Plan 17
   - 5.5 Personnel Resource Requirements 18
   - 5.6 Resource Requirements 18
   - 5.7 Costs 19
6. Project Risks & Known Issues 20
7. Project Constraints, Assumptions and Interfaces with other Projects. 21
8. Project Organization 22
   - 8.1 Stakeholders 22
     - University of South Australia 22
     - UniSA - School of Electrical & Information Engineering 22
     - UniSA- Students 22
   - 8.2 Steering Committee 22
   - 8.3 Core Team 23
   - 8.4 Other Teams as needed 23
9. Project Governance 24
   - 9.1 Meeting Structure 24
   - 9.2 Progress Tracking 25
   - 9.3 Risk & Issue Management 25
     - 9.3.1 Risk Analysis and Mitigation Path 26
     - 9.3.2 Risk Matrix 27
10. References 28
11. Bibliography 29

1 Executive summary

The Electronic Engineering department of the University of South Australia builds and develops Robots and Robotic applications for a vast range of proof of concept and real world applications.

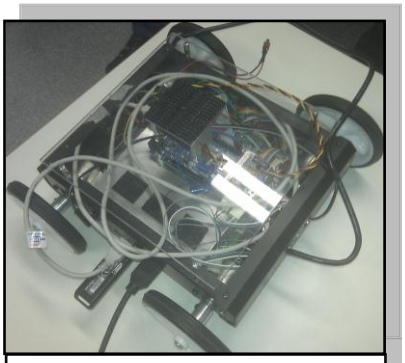

Figure 1: A FitPC/ROS based Robot (Smith, C)

The target audience of the robots, many similar to Figure 1, range from introducing high-school students in to the world of electronic engineering and robotics all the way to world class research and development teams.

This work is greatly simplified through the use of the Robot Operating System [1] (ROS) – an Open Source Operating System which among many other advanced advantages and features, adds the ability to distribute processing across many ROS-enables devices (including desktop systems).

Many of these current ROS-based prototype devices owned by the University of South Australia are based on Fit-PC 2 [2] systems (Figure 2), small form factor ARM based computers, while also making use of an [3] Arduino for closer to real-time processing of I/O.

Figure 2: *Fit-PC Size Comparison* (Jennings 2009) shows a Fit-PC2 computer, similar to the computer used in the Figure 1 robot.

There has also been interest in replacing the existing Fit-PC2 computers with PandaBoard [4] based systems in future prototypes, as PandaBoard devices are not only cheaper but also is better supported when only using non-proprietary drivers.

The Open Computer Vision [5] (OpenCV) library is also used extensively to perform tasks such as facial recognition and identification as well as object analysis and even obstruction/path finding.

2 Purpose of the Project Initiation Document

The Project Initiation Document of the ROS GUI Project will summarize all important information about the project. The Project Initiation Document is an orientation, framework and guideline for the ROS GUI Project.

The Initiation Document will describe

- What the project will deliver
- How the project will deliver
- When the project will deliver

The Initiation Document contains the following information:

- Objectives & goals of the project
- Scope
- Project approach, schedule
- Project organization, roles & responsibilities
- Project governance, processes

3 Competitive Intelligence Support

3.1 Purpose of the project

We have been asked to plan, design and implement a Graphical User Interface (GUI) which supports remotely interfacing with ROS devices, and the ability to interpret data with the OpenCV image processing library from one or more remote cameras. The application should work on both GNU/Linux [6] and Microsoft Windows Operating Systems. It has been requested that all deliverable code is licenced under an Open Source [7] licence.

The target audience is the same wide range of people that are using current technologies. This includes high-school students who may be interested in basic functions such as the ability to 'drive' the robot around while watching the live webcam feed, all the way to researchers and PHD candidates who will want to add extremely complicated custom functions which could potentially become independently published applications.

As a result, the application should be as simple to use as possible while simultaneously being easy to add new functionality or make modifications to existing works. This will be achieved through the use of high quality tutorials and documentation later discussed in 3.1 – Deliverables.

It has also been requested that an Android [8] counterpart be developed if time permits. The application should mimic the functionality of the desktop application, as well as integrating portable device features such as accelerometers for navigation. It is accepted that the Android counterpart may be limited because of the slower processing speed and library compatibility, but at the very least the ability to send and receive data is expected.

There has also been some suggestion of a commercial robotics package to fit into the intermediate market of robotic solutions, as a niche market exists for robotic kits that fit in the sub-$10,000 price range. While the deliverables of this project themselves would be freely obtainable, packaging a robotic kit with a supporting graphical interface would increase its chance of success significantly.

There is an Interest in source code for deliverables to be released under a 'free' or Open Source license such as the Berkeley Source Distribution (BSD) [9] licence, which is predominately used for ROS development. It is understood that any license which enables entities the ability to legally modify and share source code will be acceptable and as such, the General Public Licence (GPL) [10] or Lesser General Public Licence (LGPL) [11] may be used if necessary.

3.2 Benefits

The modifications planned for the ROS system will significantly reduce the amount of time required during the hardware and software testing process, this is due to the reduced time and complications required to test basic functionality as well as the ability to remotely execute functions.

Additionally it will allow for teaching students about far more complicated concepts or tasks while abstracting them away from underlying complications, this will not only enable students to learn about more complicated concepts, but will potentially increase interest as easy-to-use interfaces have proven to be effective in many other products.

The application will also allow for modifications to be made, which means it could potentially be invaluable for advanced development of solutions by students, teachers and researchers.

Releasing code under a 'free' license such as the Berkeley Source Distribution (BSD) would increase uptake of the application by companies, education institutions and individuals, contributing to a wider advancement and uptake for the purposes of research and improvements in robotics. Not only does such an achievement increase technology globally, but it also allows all stakeholders and contributors to get credited for such an achievement.

Major improvement themes

- Ability to create more advanced functionality with a supporting GUI quickly and easily
- Faster development time
- Decrease knowledge required to operate robots
- Reduces development and testing during the building and development of robotic devices and software

Major impact themes

- Reduced research & development costs
- Reduced training costs
- Education quality improved due to faster uptake of ROS
- Revenue increase through higher student exposure to easy-to-use robots, resulting in increased course uptake
- Increased identification of UniSA's robotics department worldwide

3.3 Limitations

Development Language

It has been requested that Python 3 [12] be used unless specific reasons exist which prevent this. This is a likely option as Python appears to be one of the preferred languages in the ROS community, and apart from native C libraries appears to be the only language with widely available ROS wrapper libraries.

Time

The project is expected to be completed on the 9th of November 2012, which is a reasonably short period of time. As each student is expected to work approximately 16 hours per week (University of South Australia 2010) on the project this equates to about 5 weeks full time equivalent per person.

Budget

Although no exact budget has been specified, it is expected that the entire project will require a budget of no more than $500. This includes the (potential) purchase of an Android phone or Tablet computer.

Licensing

As it is preferred that the application code is released under the Berkeley Source Distribution (BSD) license or the General Public Licence (GPL) and Lesser General Public Licence (LGPL) if it is deemed a requirement, careful attention will need to be paid to which license is used for any code that we include in our project. If there is as inclusion of any strict-licenced code, deliverables may be prevented from being released under a more lenient licence, which in the case of ROS development shall be the BSD License as it has been requested as the open source license of choice for this project.

Graphical Interface

It is expected that the PC version of the application should work correctly both on GNU/Linux and Microsoft Windows systems, preferably without the use of software such as Cygwin [13] or WINE [14]. Not only will this limit the selection of the GUI engine, but will also increase testing time. While this limitation exists, there are several GUI engines that support both GNU/Linux and Microsoft Windows such as GTK+ [15], QT [16], TK [17] and many others which will still be suitable candidates.

3.4 Critical Success Factors

Client Satisfaction

As the application's success is heavily dependent on the satisfaction of the client, the client will be heavily involved in the design of both the GUI and any expandable framework. The clients' satisfaction with the final product will be measured through the use of a qualitative survey form.

Application Functionality

The final application must function as outlined in the requirements stated previously. The application must work on both Microsoft Windows and GNU/Linux, and feature a graphical interface which allows sending and receiving commands and responses to the robot.

Application Interface

The GUI should be easy to use, straight forward, and concise to the point where a young adult with only basic knowledge of computers should be able to operate it, including the loading and execution of third party modules. Configuration will require more knowledge of GNU/Linux, Microsoft Windows and ROS as well as a basic understanding of networking fundamentals. It is assumed that most students of the Electronic Engineering department would be capable of configuring the application with little or no formal training.

Quality of Coding / Developer Documentation Styles/Standards

It is expected that the application we release is well coded, commented and documented. Tutorials will be provided with the intention of teaching users the steps required in order to add or modify features and functionality, and must be easy to understand and follow. We will follow the REP8 (Conley 2010) and PEP8 (Van Rossum & Warsaw 2001) Style guides for Python, as well as implementing API documentation in Epydoc [18] or a similar alternative.

4 Scope

4.1 In Scope

Application Functionality

The application will be reasonably minimalistic and feature only movement controls and the ability to stream video and apply OpenCV based filters to the stream.

While the Application will have few user-facing features, it will be designed with expandability in mind; developers who wish to add or modify features should be able to do so quickly and easily.

Application Usability

It is expected that Users should still have a basic understanding of ROS, and be capable of basic tasks on Microsoft Windows, GNU/Linux and Android platforms in order to configure systems prior to execution of project deliverables. However, once set up, it is expected that anybody familiar with point and click interfaces will be capable of remotely controlling a robot.

Application Expandability

Anybody who is familiar with the basics of the python programming language should be capable of expanding the functionality of the application after following the development tutorial and documentation (discussed in the number in the deliverable section).

4.2 Out of Scope

Application Functionality

The initial application will not include any extensive functionality further than a remote control/movement option, the implementation of basic OpenCV filters to the video stream as well as the ability to communicate messages between the robot and the user interface.

Application Usability

It will be expected that all users of the software understand the basic concepts of computer systems. This should fit the entire targeted demographic for the immediate and foreseeable future.

Application Expandability

It is expected that users who wishes to make modifications to the Application will have existing knowledge of Python and ROS, as well as familiar with the documentation provided as a deliverable.

Security

It is understood that ROS currently offers very little internal security. Many of these issues cannot be fixed without extensive knowledge of the internal workings of ROS and modification to the existing protocol. For that reason, the security of any deliverable code cannot be guaranteed. It is recommended that ROS nodes are configured to use an SSL tunnel or some other form of point-to-point cryptography on top of the existing encrypted wireless network if a secure environment is required.

5 Project Plan

5.1 Project Approach

The project began on 23rd July 2012 and must be completed by the 9th November 2012. After this time the project will be licenced and released under on Open Source licence, allowing anybody who wishes to continue the project to maintain it.

5.2 Project Schedule & Milestones

The Project Milestones in order to reach this goal is as follows:

Task Name	Duration	Start	Finish	Predecessors
Write Project Charter	16 days	Mon 23/07/12	Mon 13/08/12	
Research ROS	**16 days**	**Fri 3/08/12**	**Fri 24/08/12**	
Generic research and Understanding	12 days	Fri 3/08/12	Mon 20/08/12	
Figure out how to setup ROS to run in a distributed environment	**2 days**	**Thu 23/08/12**	**Fri 24/08/12**	**3**
Research and Testing	2 days	Tue 21/08/12	Wed 22/08/12	3
Write Distributed ROS Environment Tutorial Deliverable	2 days	Thu 23/08/12	Fri 24/08/12	5
Generic ROS-Python module Understanding	2 days	Thu 23/08/12	Fri 24/08/12	3,5
Research OpenCV	**8 days**	**Tue 21/08/12**	**Thu 30/08/12**	**13**
Interface camera with OpenCV on local system	3 days	Tue 21/08/12	Thu 23/08/12	
Implement webcam streaming with OpenCV support	**6 days**	**Thu 23/08/12**	**Thu 30/08/12**	
Research / Testing	3 days	Thu 23/08/12	Mon 27/08/12	
Write tutorial document (May not be required)	3 days	Tue 28/08/12	Thu 30/08/12	11
Design of modular application prototype	**7 days**	**Thu 16/08/12**	**Fri 24/08/12**	**1**
Research	2 days	Thu 16/08/12	Fri 17/08/12	
Development	2 days	Mon 20/08/12	Tue 21/08/12	14
Discussion with Client	1 day	Fri 24/08/12	Fri 24/08/12	15
Design and Write final application	**26 days**	**Fri 31/08/12**	**Fri 5/10/12**	**1,2,8,13**
Planning	2 days	Fri 31/08/12	Mon 3/09/12	
Development	10 days	Tue 4/09/12	Mon 17/09/12	18
Testing	14 days	Tue 18/09/12	Fri 5/10/12	19
Write Developer Tutorials/Documentation deliverables	**15 days**	**Tue 18/09/12**	**Mon 8/10/12**	**19**
Planning	2 days	Tue 18/09/12	Wed 19/09/12	
Writing	10 days	Thu 20/09/12	Wed 3/10/12	22
Testing / Discussion with Client / Test Subjects	3 days	Thu 4/10/12	Mon 8/10/12	23
Research Android Development	5 days	Tue 4/09/12	Mon 10/09/12	1,18
Design and Write Android Application	**17 days**	**Thu 4/10/12**	**Fri 26/10/12**	**25,23**
Planning	2 days	Thu 4/10/12	Fri 5/10/12	18
Development	10 days	Mon 8/10/12	Fri 19/10/12	27
Testing	5 days	Mon 22/10/12	Fri 26/10/12	28
Buffer Time	10 days	Mon 29/10/12	Fri 9/11/12	26,21,17,13,12,1

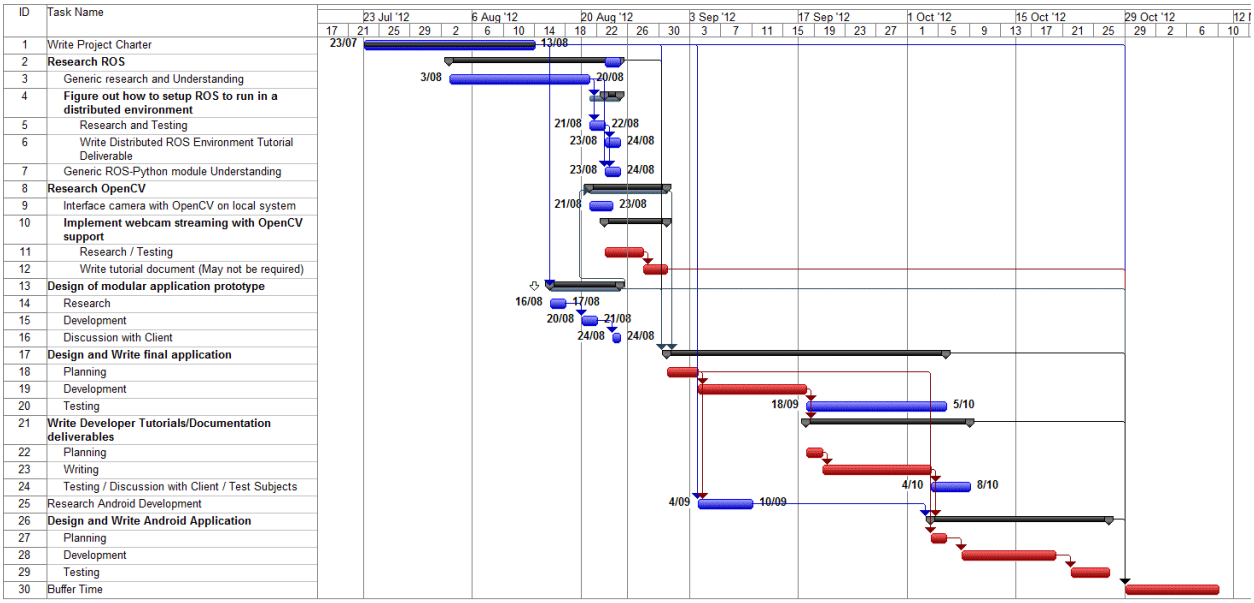

Figure 3: A screenshot of the Project Gantt Chart showing the critical path in red. The Gantt chart is provided with the project documentation in electronic form.

5.3 Deliverables

Document: ROS GUI Project Charter

The ROS GUI Charter document (this very document) will describe what, how and when the project will deliver, as well as objectives, goals, scope, schedule and potential risks.

Finish Date: Monday, 13th August 2012.

Document: How to setup distributed ROS Environment

A document detailing how to configure ROS in a distributed environment as well as how to setup ROS nodes for remote access will be produced. The document will be followed by test subjects who are familiar with ROS, GNU/Linux and TCP/IP network fundamentals to ensure the documents are comprehendible. This will be delivered in ether Microsoft Word 2010, PDF or HTML format.

Finish Date: Friday 24th August 2012

Test Code: Modular Application Prototypes

Prototype modular applications will be written to test / explore techniques for a modular approach. These will show experimental attempts at different modular framework implementations. The prototypes will be explored and discussed with the client, and a decision on which style to use will be made.

Finish Date: Friday 24th August 2012

Test Code: OpenCV Implementation

Code will be written to implement OpenCV. OpenCV will be integrated with ROS and the project GUI. Test code used / written during research will show progression in understanding of OpenCV and will enable image processing of OpenCV images to be performed on the robot platform and streamed to the GUI.

Finish Date: Thursday 30th August 2012

Deliverable: Streaming Video How to

A document describing how to install and configure software relevant to streaming video from a Webcam host to clients will be produced. The document will be followed by test subjects who are familiar with ROS, GNU/Linux and TCP/IP network fundamentals to ensure the documents are comprehendible. This will be delivered in ether Microsoft Word 2010, PDF or HTML format

Finish Date: Thursday 30th August 2012

Code: Final Application

It is expected the final application will take roughly two weeks to develop and a further two weeks of testing / debugging. The final application should be capable of connecting to a robot and communicating messages between the robot and the user Interface. Messages will include but are not limited to communications such as sending movement instructions and streaming video (with the ability to apply OpenCV filters). The application will also include loading and unloading of custom user written modules. The application will be tested by the Client, and the target audience to assure it is straight forward and easy to use.

Finish Date: Friday 5th October 2012 (Including testing time)

Documents: Tutorials / Developer documentation

Tutorials and developer documentation will be required in order to teach/train users who wish to develop custom modules for the application. It is assumed that users have some programming experience, but not necessarily Python, and only a basic understanding of ROS and OpenCV. The documents will be tested by the Client, and the target audience to assure it is straight forward and easy to use. Documents will be delivered in ether Microsoft Word 2010, PDF or HTML format.

Finish Date: Monday 8th October 2012

Code: Android Application

An Android application has been requested if time permits. The Android application is hoped to support all features from the desktop application with additional support for features such as accelerometers. Documentation regarding further development will be written simultaneously. The application will be tested by the Client, and the target audience to assure it is straight forward and easy to use.

Finish Date: Friday 26th October 2012

5.4 Communication Plan

General

In order for the development team to stay well informed several mediums of media will be used; e-mail, Facebook Instant Messenger, Skype and telephone calls as well as weekly face to face meetings will be held to keep in touch. A GIT [19] repository will also be used to ensure all users are always working with the most up to date version of the software, this is likely to be hosted on GitHub [20].

Meetings

Once a week the software development team, the team supervisor and the client will meet to discuss the project. The discussion will include how the project is progressing compared to the schedule, any unexpected delays and clarification of details, to ensure the refinement of the project as it progresses.

Development Team Communication

In order to keep everyone informed of the state of the project, the development team will also have a weekly meeting over Facebook or Skype. In addition, after editing, adding or deleting any code the software team must be informed of the changes via meaningful check-in/commit comments.

Android Interface Development

The development team will need to meet in person to design the Android application interface. This will be arranged when the time is necessary.

5.5 Personnel Resource Requirements

It is estimated that the project will require three part-time graduate developers working for approximately 20 hours per week. Regular consultation with a supervisor is thought to reduce the development time significantly, it is estimated about 20 hours of supervisor time will be required total.

5.6 Resource Requirements

Lab Space

As we currently have no office, a space for our team to develop collaboratively will be required. This area should be spacious, with computer desks, office chairs, power points for Laptop computers, and access to an Ethernet / wireless network.

Development Computers

It has been requested that two dedicated research and development machines be provided onsite. These systems must meet our minimum requirements of being Dual Core systems with at least 2GB of RAM and 200GB of disk space which correctly operate in both Microsoft Windows and GNU/Linux environments. We are also require Administrative privileges for the local system as it is likely these will be required for installation of development software and ROS.

Bandwidth

Many software components are freely available on the internet, since some packages exceed upwards of 1GB, we will require up to 100GB of bandwidth per month and speeds above 3Mbps.

Android Device

As an Android application has been requested, an Android device will be required in order to develop the software. The device should have Accelerometers, and an 802.11b/g/n compatible wireless interface. Our recommendation is the Google Nexus 7 [21]. We estimate we will require the device by 29th September 2012 until 22nd October.

Test Robot Device

We will require a Robot for testing the final application. We expect this to be of no more than four periods of three hours. Times which they are not required by others will be discussed with the Client when necessary.

Webcams

The use of a Webcam will be required in order to develop the video streaming and OpenCV counterparts of the application. The webcams are required to be supported and stable under a GNU/Linux environment. We expect these will be required throughout the entire project, however it is believed the Client has several webcams that we can use for this period of time.

5.7 Costs

Code	Description	Theoretical Cost	Actual Cost
Personnel	3 x Graduate Developers @ 50kPA for 5 Weeks FTE, 1 x Supervisor / Senior Developer @ 120kPA for 20 Hours.	$15,576	Not funded
Snacks and Beverages	High sugar and caffeine products.	$150	Not funded
Development Computers	2 x $1,250 computer systems	$2,500	N/A (Provided)
Bandwidth	Approximately $70 / Month over 3 months (Based on the suggested bandwidth requirements from section 5.6 above)	$210	N/A (Provided)
Android Device	Google Nexus 7	$249	$249
Webcams	2 x GNU/Linux compatible Video Cameras for use during development @ $60 each	$120	N/A (Provided)
Test Robot	Test robot to ensure application works as expected. Used for short periods of time.	Unknown	N/A (Provided)

6 Project Risks & Known Issues

The following table represents risks that could potentially harm the project. It contains the likelihood of the risk expected and the potential worse case impact of the risk if it were to happen.

Risk ID	Risk	Likelihood	Impact Description
1	Change of requirements in the project or features added along the life span of the project.	Likely	It is possible that the requirements could change, causing minor delays in the project schedule.
2	Absence of Supervisor / Client during meetings and project development.	Likely	If the supervisor is absent from meetings it could result in absence of external guidance when required. If the Client is not available it could potentially cause major misdirection in the project.
3	Loss of sponsor due to redirection of UniSA and its resources.	Unlikely	If the sponsor were to pull out, it would mean that the project would have to be scrapped and resources would be wasted.
4	Failure to meet schedule in a suitable time frame.	Likely	It is possible that the project will not be finished on time which would have a high impact on the project in terms of team satisfaction and time wastage for all parties involved.
5	Differing opinions on the final project design caused by group malfunction.	Likely	It is likely that members of the team may not see eye to eye on different stages of the project, causing issues within the team and decreasing motivation to finish the project.
6	Project differs from clients expectations due to miscommunication.	Likely	It could be likely that at the end of the project, the final deliverable is not what the client wanted which would impact the project negatively in terms of wasted resources and an unsatisfied client.
7	Unable to test software on a physical robot in a suitable time frame due to delays in a test robot.	Unlikely	It could be likely that when it is time to test the software on a physical robot, other required teams may not have a robot ready to test on. This could impact the project by causing delays.
8	Loss of team members due to unforeseen circumstances.	Unlikely	If the amount of team members were to change it would cause delays to the project.

7 Project Constraints, Assumptions and Interfaces with other Projects.

Constraints

- All software dependencies must support Microsoft Windows, GNU/Linux
- The project needs to be complete by 1st November, 2012 as this is the end of Study Period 5 for the University of South Australia
- The Android device is expected to be a midrange device costing no more than $300-$400 AUD.
- All direct code inclusions will constrain the licence of the final deliverable.

Assumptions

- The Application will need to interface with both ROS and OpenCV. Python wrappers for both exist, it is assumed that these are feature-complete, stable and receive continuous development & support.
- It is assumed that we will not be required to retrofit existing UniSA-owned ROS projects or programs or their functionality into our application.
- It is assumed that all Robots will be GNU/Linux based systems
- Anybody who is required to connect to a remote system is assumed to be able to obtain IP address or Hostname without help, or be provided with the details.

Interfaces with Other Projects

- The final application will not have a direct interface API, however extending its functionality will be possible with aid from the tutorials and documentation.
- Although technically not an interface, all code will be well documented internally, allowing anybody who wishes to further extend functionality, or adjust code for any other reason, they will be able to.

8 Project Organization

8.1 Stakeholders

University of South Australia

The University of South Australia will benefit from the project by utilising the software package created in this project as a learning aid for students. UniSA will be able to use the software in any course where communicating with robots is required.

UniSA - School of Electrical & Information Engineering

The school of Electrical & Information Engineering will be utilising the finished project in a new course that will run in 2013.

UniSA- Students

The final software will be used to assist the students understanding of ROS while also allowing students to focus their attention on building robot hardware components rather than the details of this project.

8.2 Steering Committee

The steering committee has the following members:

- **Dr Russell Brinkworth (Project Owner)**

  Dr Russell Brinkworth is the project owner who has requested the final project that shall be used in the development of a new course. This course will be developed for the School of Electrical and Information Engineering and will serve as a teaching aid to students at all levels.

- **Graeme Robertson (Project Owner)**

  Graeme Robertson is in charge of putting together an affordable robot package that will be used with the final project deliverable. The final software package built in this project may be bundled and used with the robot hardware package put together by Graeme.

- **Dr Ivan Lee (Supervisor)**

  Dr Ivan Lee will serve as the supervisor of the project and will help the core team wherever needed in order to ensure that the final project is delivered on time and to a standard acceptable by the project owners.

8.3 Core Team

The core team has the following members:

- Colin Smith (Team Lead, Programmer)
- Matthew Babiszewski (Programmer)
- Jason Kerslake (Programmer)

8.4 Other Teams as needed

Other teams will be required as the project progresses such as those involved in building and supply of robots that will be utilized with the project.

These teams will be involved during the course of the project and will be able to supply robots as the project requires them. The timing of the use of the other teams will be governed by final testing requirements of the software with the actual target robot for the final project deliverable.

9 Project Governance

9.1 Meeting Structure

The **Matrix of Responsibility** below describes the inner workings of the team in terms of who is in charge of a certain responsibilities of the project, who will lead that role and any supporting roles.

Responsibility	Project Owner	Supervisor	Team Leader	Programmers	Supporting Teams
Ensure project plan is acceptable	L	S			
Ensure that the core team is provided with support	S	L			
Track progress of the project		S	L	S	
Ensure Team Leader is supported throughout project			S	L	
Ensure the team functions in a productive manor		S	L	S	
Ensure all sub tasks are completed on time		S	L	S	
Ensure final deliverable is complete and on time		S	L	S	
Ensure project deliverables are what the client requested		S	L	S	
Ensure code is to a good standard and quality			L	S	
Ensure code is documented in an easy to read manor for final deliverable			L	S	
Ensure physical testing robots are available.	L				S

L = Lead Role S = Supporting Role

9.2 Progress Tracking

The weekly stand up meetings will be used as a checkpoint to track the progress of the project. These meetings will consist of representatives of all parties involved in the project including members of the steering committee and the core team.

The meetings will be used to help track project tasks and ensure that all task deadlines are met and on schedule. The meetings will also be used to track the status of the project and help alleviate any difficulties to ensure that the final project deliverable is on schedule, complete to specification and delivered on the due date.

A Gantt chart will also be used in order to track actual due dates and milestones to ensure that the project is on schedule as well as helping the team to know where they should be and how far along the project is from completion.

9.3 Risk & Issue Management

The potential risks from section 5 have been assessed in the Risk Analysis and Mitigation Path table in section 9.3.1 which is found on the next page for presentation purposes and a 'Risk Level and Score' for each risk has been calculated using the risk matrix in section 9.3.2.

9.3.1 Risk Analysis and Mitigation Path

Risk ID	Likelihood	Impact	Risk Level & Score	Mitigation Plan
1	Likely	Minor	Low - 1	As change of requirements is likely, a significant amount of slack time will be allocated. It is expected that any change in requirements will not be any more significant than slight modifications in application behaviour.
2	Likely	Moderate	Medium - 2	Our Supervisor Dr Ivan Lee has advised that he will be away from Week 9 onwards, however he has arranged a replacement to supervise the project and the team will have to get the new supervisor up to speed.
3	Unlikely	Major	Medium - 2	As the project is working for an internal branch of the University of South Australia, and functionality of the project expected to substantially benefit a course beginning next year, it is unlikely the sponsor will discontinue the project or cease communication as a body.
4	Likely	Major	High - 3	Failure to meet schedule will be prevented by good management and planning along with regular group meetings. The team will ensure that the time and checkpoint schedules are met by using a Gantt chart to track progress where slack is provided on the critical path. This will ensure that any problems or issues that are faced are able to be mitigated.
5	Likely	Minor	Low - 1	Team work, strong lead and direction will be used as the mitigation plan and any members that are unhappy will be able to discuss any issues with the project supervisor.
6	Likely	Moderate	Medium - 2	The team will be using this document to gather specific requirements of this project and the requirements will be confirmed with Dr Russell Brinkworth to help alleviate this risk.
7	Unlikely	Moderate	Low - 1	This risk should be minimal as Dr Russell Brinkworth will ensure a robot is available at the time of need. As the software should be able to run on a range of non-specific devices and robots, this risk will be minimized.
8	Unlikely	Major	Medium - 2	The remaining team members will have to pick up the slack as this risk won't be able to be avoided if the team should lose a member and they will have to finish the work on time.

9.3.2 Risk Matrix

The values in columns labelled 'Likelihood' and 'Impact' in the risk identification plan above, respectively refer to the likelihood measure on the Y-axis and the Impact measure along the X-axis of the risk matrix presented below. It is from these two input values that the risk level and therefore risk score can be derived from the risk matrix presented below.

< Likelihood >

Very Likely	Medium 2	High 3	Extreme 4
Likely	Low 1	Medium 2	High 3
Unlikely	Low 1	Low 1	Medium 2
	Minor	Moderate	Major

< Impact >

Risk Level Description

A general description of the risk level is presented in the table below and explains in words, the potential impact of the risk as derived from the risk identification plan and found in the risk matrix above.

Risk Score & Risk Level	Risk Impact
1 - Low	Risk is negligible. Mitigation Plan could avoid impact of risk or very unlikely chance of happening.
2 - Medium	Risk has a chance of happening and could cause setbacks in the project. Mitigation plans must ensure risk can be avoided as much as possible.
3 - High	Risk is highly possible and a mitigation plan must attempt to cover all potential undesired outcomes thoroughly.
4 - Extreme	Risk will definitely happen. The most extreme case. Mitigation Plan(s) will need to reduce these risks where possible or an alternate option should be discussed.

References

Jennings, M 2009, *Fit-PC Size Comparison*, viewed 11 August 2012, <http://photos.pcpro.co.uk/blogs/wp-content/uploads/2009/06/fitpc1.jpg>.

Van Rossum, G & Warsaw, B 2001, *PEP 8 -- Style Guide for Python Code*, viewed 13 August 2012, <http://www.python.org/dev/peps/pep-0008/>.

Conley, K 2010, *Style Guide for Python Code*, viewed 13 August 2012, <http://www.ros.org/reps/rep-0008.html>.

University of South Australia 2010, *Time/workload management*, viewed 13 August 2012, <**http://w3.unisa.edu.au/counsellingservices/balance/workload.asp**>.

Bibliography

[1] ROS (Application) – A Robotic Operating System & Framework, Various Authors
http://www.ros.org/wiki/

[2] Fit-PC (Hardware) – A small form factor ARM based computer, Designer unknown
http://www.fit-pc.com/web/

[3] Arduino (Hardware schematics) – Open Source schematics for a flexible controller, Various Authors
http://www.arduino.cc/

[4] PandaBoard (Hardware) – A competing small form factor ARM based computer, Designer unknown
http://pandaboard.org/

[5] Open Source Computer Vision (OpenCV) – An Open Source library for programming functions for real time computer vision, various authors
http://opencv.willowgarage.com/wiki/

[6] GNU/Linux (Software) – An Open Source Operating system compilation, Various Authors
http://www.gnu.org/gnu/linux-and-gnu.html

[7] Open Source (Software Philosophy) – A philosophy that application source code should be obtainable and distributable without restriction, Author not applicable
http://en.wikipedia.org/wiki/Open_source

[8] Android (Application) – A Mobile Operating System developed by Google, Various Authors
http://www.android.com/

[9] Berkeley Source Distribution (BSD) Licence – BSD License Definition, various contributors
http://www.linfo.org/bsdlicense.html

[10] General Public Licence (GPL) (Licence) – A 'Free as in speech' licence, various contributors
http://www.gnu.org/copyleft/gpl.html

[11] Lesser General Public Licence (LGPL) (Licence) – A licence similar to GPL with fewer restrictions on modified work than GPL
http://www.gnu.org/licenses/lgpl.html

[12] Python (Programming Language) – An Open Source Object Oriented JIT programming language, Various Authors
http://www.Python.org

[13] Cygwin (Application) – A GNU/Linux like environment for Microsoft Windows. Various authors
http://www.cygwin.com/

[14] WINE (Application) – Run Microsoft Windows dependant applications on GNU/Linux, Various Authors http://www.winehq.org/

[15] GTK+ (GUI Toolkit) – The GIMP+ Toolkit allows creating Graphical User Interfaces, Various Authors http://www.gtk.org/

[16] QT (GUI Toolkit) – A cross-platform application and UI framework, Various Authors http://qt.nokia.com/

[17] Tk (GUI Toolkit) - A basic cross-platform graphical user interface, Various Authors http://www.tcl.tk

[18] Epydoc (Python Library) – An API Documentation generator for Python, Various Authors http://epydoc.sourceforge.net

[19] GIT (Software) – A branching source control system, Various Authors http://git-scm.com/

[20] GitHub (Software as a Service) – A free-to-use GIT repository service for Open Source projects https://github.com/

[21] Google Nexus 7 (Tablet Computer) – A low-cost Tablet PC designed by Google http://www.google.com/nexus/#/7