**Software Requirements Specifications**

**NASA / Department of Mechanical Engineering**

**Planetary Rover Control Software**

**Prepared by: Daniel Bittner, Edward Carter, Jun He, Matthew Ng, Dakota Pollitt, Zhentao Zhong**

**Class: Software Engineering – Fall 2014**

**Instructor: Dr. Rusu**

**Table of Contents**

1. Introduction………………………………………………………………………………. 3
2. Executive summary………………………………………………………………………. 4
3. Application Content……………………………………………………………………… 4
4. Functional Requirements………………………………………………………........…….5
5. Environmental Requirements………………………………………………………...….. 5
6. Training……………………………………………………………………………………6
7. Software Qualities………………………………………………………………………....6
8. Other Requirements……………………………………………………………………….8
9. Time Schedule……………………………………………………………………….……8
10. Potential Risks…………………………………………………………………….………8
11. Future Changes…………………………………………………………………..………11
12. Acceptance Test Plan………………………………………………………….…………12
13. Glossary…………………………………………………………………………….....…13
14. Prototype Captures…………………………………………………………….…………13
15. Reference Documents……………………………………………………………...…….14
16. **Introduction**

The Rowan University Engineering team for the NASA Robo-Ops competition has requested assistance with the software aspects of their Mars Rover. The purpose of the Rover as a whole is to traverse multiple maps and search out certain color rocks while being controlled remotely from another state. The user, located in NJ, will have video feeds from the Rover located in Texas and be able to send commands using an Xbox controller.

These commands allow the user to control the robots movements and views. We have been asked to focus on the video feeds and the communications, as well as developing a client-side GUI application.

**System Overview:**

*This document contains the following sections:*

* Introduction
* Executive Summary
* Application Content
* Functional Requirements
* Environmental Requirements
* Training
* Software Qualities
* Other Requirements
* Time Schedule
* Potential Risks
* Future Changes
* Acceptance Test Plan
* Glossary
* Prototype Captures
* Reference Documents

1. **Executive Summary**

The RASC-AL Robo Ops competition has been held numerous times before, and many products and reports are available for research. The Rover will have six wheels which are controlled by ten motors for high traction and simplicity. To turn, the inside two wheels will act as pivot points while the other four wheels will turn. The Rover will be mounted with an eight camera system with image processing and possibly autonomous navigation ability. The user should be able to control the rover with an Xbox controller or joystick and view the video stream from the cameras over a wireless network. This system will implement the following features:

* Users will be able to toggle through the Rover’s various video feeds
* The GUI will display at least 2 video feeds simultaneously
* Users will be able to drive the Rover through various obstacles
* Users will be able to operate the Rover’s Articulating Arm
* User will be able to remotely control the Rover and its Articulating Arm using a 4G network

Important risks to be taken into consideration

* **Wireless Communication:** Any interruption in wireless connection will render the Rover inoperable. The Rover may need to make use of two wireless networks.
* **User Interface:** The GUI must be quick and smooth to allow easy viewing. This is how the driver will decide where the Rover is and where it needs to go. Any GUI faults could cause the drivers to be ‘blind’ and not control the Rover.
* **Controls:** The controls need to work and have fast response times. Control errors could cause the Rover to become immobile.

1. **Application Content**

* The GUI will have at least 2 active video feeds on the top half
* The bottom half of the GUI will display a model Rover that shows which parts are moving from a given control
* The Left Analog Stick/Directional Pad will be used to control the Rover’s movement
* The X and Y buttons will be used to toggle the individual camera feeds. Extra potential feeds will be assigned to the A and B buttons
* The Right Analog stick will allow free movement of the wheels
* A User with a second controller will operate the Articulating Arm
* The second controller will be able to move the Articulating Arm laterally and vertically
* The second controller will be able to manipulate the Articulating Arm gripper to grab and release objects

1. **Functional Requirements**

**4.1 Video Feeds**

**4.1.1** After opening the GUI, a toggle button can be used to switch between viewing the multiple cameras from the Rover.

**4.1.2** Multiple video feeds can be viewed at a time from base command.

**4.2 Robotic Controls**

**4.2.1 Move Forward -** Allows the Rover to move forward at various speeds.

**4.2.2 Move Backwards -** Allows the Rover to move backwards at various speeds.

**4.2.3 Free Turning** - Allows free rotation of the wheels for gradual turns

**4.2.4 Turn Left** - Allows the Rover to turn left in place

**4.2.5 Turn Right** - Allows the Rover to turn right in place

**4.2.6 Arm Control** - Allows the Rover to move its Articulating.

**4.3 Communications**

**4.3.1** Data can be sent over and read through the 4G network to control the Rover remotely.

1. **Environment Requirements**

**5.1 Software Requirements**

This software will be run with the most current version of Java (Java Version 7) and C. Communication software will be installed to send data between the home base and robot. This software will currently be compatible with Windows 7 and Unix operating systems.

**5.2 Hardware Requirements**

The Rover PC must have a powerful video card to handle the number of feeds it must process. It must also have sufficient memory for the various data communication operations it must complete. The Command PC must also have a sufficient video card to process and display the feeds. The network adapter must be able to handle sufficient bandwidth for both video sending/receiving and command sending, also it has to make sure it sends out a static signals.

1. **Training**

Our team will constantly work with the customer regularly during the software creation process, and learn all the functionalities of the rover. As the GUI is designed, the customer will see how the controls work as well as be able to change what they deem appropriate. When the software is finalized, there will be a final meeting to go over all of the specifics.

1. **Software Qualities**

**Reliability:** We will be sure to thoroughly test all software before our due date to ensure all robotic functions work properly and as expected.

**Correctness:** We will be sure to include all features and functions agreed upon between both teams.

**Performance:** The systems we create will be as efficient as possible so the software runs as quickly as we can make it.

**Maintainability:** The Rover may require maintenance during the competition and afterwards for future years.

**Timeliness:** Our software will be delivered on time. This way the Engineering team can continue their project to complete the Rover.

**User Friendliness:** The Rover controls and GUI will be easy to use and functional so commands can be sent and video feeds seen.

**Robustness :** The software will be able to handle multiple commands and extended usage as needed.

**Understandability:** Proper documentation will be created to specify all aspects of the software. Any coding done will be clean and commented.

**Reparability:** Code will be clean and commented to allow programmers to easily make necessary changes.

**Productivity:** Our teamwork and team process will ensure we meet our customers expectations and deadlines.

**Evolvability:** Our software will fulfill user specifications and allow changes/upgrades to be made easily in the future.

**Re-Usability:** Our software will be usable for future applications.

**Portability:** The coding will be usable for other applications in its language. Various parts may be written in different languages.

**Verifiability:** The Rover will act upon data and commands being sent to it. It will respond and send data back to the user as well. No data should be changed while moving between the Rover and user.

**Safety:** Control of the Rover is limited to the User. This prevents outside sources from taking any unwanted control.

**Size:** The application size will be kept small and simple to allow it on any machines necessary to control the Rover.

**Interoperability:** The application will run on any computers that support java and C programs in general.

**Visibility:** The GUI and Robotic Controls will be nice to look at. Various video feeds will be able to be seen. The controls will be functional.

1. **Other Requirements**

* The customer must provide adequate Arduino programs for the arm and wheel servos
* The customer must provide adequate on board hardware for the software to perform on
* Code will need to be written in either Java, C, C++, or C#
* The customer must specify hardware needed for communication

1. **Time Schedule**

The following deadlines have been set by the team and by the client. The deadlines will ensure that the project is completed in a timely fashion and by the required specifications stated in this document. Christopher Contrevo, as well as the rest of the engineering team, will be updated frequently through text, email, and other open communication channels, as well as frequent meetings.

*The following time schedule is tentative and may be added upon at a later date*

Phase 1: The Software Requirements Specification document and prototype are to be completed by Wednesday October 8th, 2014.

Project Proposal: The Project Proposal is to be completed by Wednesday October 8th, 2014.

Phase 2:

Phase 3:

1. **Potential Risks**

Personnel Shortfalls - There is a risk that team member tasks will not be assigned based on a given team member’s strengths preventing them from completing the task on time.

Unrealistic Schedules and Budgets - The choice of rover equipment and hardware needs to be within acceptable cost constraints and the goals of the software need to be achievable within scheduled time parameters.

Developing the Wrong Software Function - The team needs to be sure that functions are built that work with the chosen rover hardware.

Developing the Wrong User Interface - There is a risk that the user interface will not display all the information necessary in order to properly control the robot or access the robot’s information.

Gold Plating - Our team needs to make sure priority is given to the network and communications between the rover and base command in order to ensure proper rover control. Focus needs to be given priority to that as opposed to superfluous functions.

Continuing Stream of Requirements Changes - Since the team will need to have constant communication with the Engineering team during software creation, there will be the temptation for the Engineering team to keep proposing changes to the software.

Shortfalls in Externally Furnished Components - The majority of the hardware that will be running our software will be furnished by the Engineering team so there may be failings in the equipment provided.

Shortfalls in Externally Performed Tests - If a prototype rover is not completed the time of software testing, there is a risk that Engineering team might not be able to test the software within test plan parameters.

Real-Time Performance Shortfalls - The rover will need to be able show video feed in real time as well as follow movement commands in real time. If either of these component modules fail to perform adequately in real time, the entire rover will fail.

Straining Computer-Science Capabilities - The project should not require any software capabilities that would not already be possible by networked robot controls.

Requirements - The requirements will be laid out so that there is no risk of misunderstood software capabilities or misunderstanding about what the team needs to accomplish.

Technology - The technology and software needed for developing the rover is available at this time.

Business - There should be no business risks for this project since the software will not be used commercially.

Political - There is no political risks for this project

Resource - If our proposal did not make it into NASA’s competition, we might need to get funding from the engineer department. Also we need a prototype of the rover to test out the software.

Skills - Our team will be using unfamiliar hardware in regards to networking. We will also be using programming languages such as C that the team only has limited experience in.

Deployment - The software for both controls and communication must work properly with almost no bugs. If the software is the fail, the users may not have any robotic visuals or control over the robot.

Integration - There is a risk that the system choices for the rover PC and motor control systems will not work with the software and the commands being issued to the rover when the software is integrated.

Schedule - The schedule of developing the software isn’t really clear at this time, since this is a two semesters project, and we are only having one semester to work on what we can complete.

Maintenance - Our team must document all parts of the software so it can be easily maintained by us and future developers. Since this is a rather large project it is likely that a different team of developers will take over after us. Therefore it is imperative that we properly comment and document all code to make it easy to maintain.

Design - We must further discuss the design of the project and have clearly defined requirements. Since the rover is not yet completed we risk our design not being compatible with the final rover. The design of the user interface must also be clear and easy to use otherwise the user will not be able to control the rover well. The prototype below shows an example of what the user interface will look like.

Development of hardware and software choices will require constant communication with the Engineering team. Failures in communication could cause the team to work on developing functions and software that will not work correctly within the rover’s design.

1. **Future Changes**

When our team completes the software piece given to us for the Mars Rover Project, the robot will not yet be complete. Many changes and updates will be added to the Rover before it is ready for competition. Some possible changes to be done are as follows.

**Upgraded Displays**

More video feeds from the robot can be added to allow easier viewing from the control team.

**Autonomous Software**

Software can be added to the robot to allow it to perform safety checks while moving or move on its own to a given location. Also software can be updated so the rover has the ability to perform self driving when there is no human driver or lost communication between human driver and the rover.

**Upgraded communication**

The hardware and software used to communicate between the control station and robot can be upgraded to allow quicker communication speeds.

**Upgraded CPU**

The CPU of the rover can be upgraded for faster calculation when autonomous software is added to the Rover.

**Add more driving modes**

During different environment of the rover driving, it requires different wheel power and speed to drive over obstacles or climbing uphill or etcs.

**Controller upgrades**

The controller and buttons used to communicate with the robot can be changed or moved over time.

**Coding changes**

The code is easily accessible to change, fix, or upgrade as time goes by.

**GUI changes/updates**

As more physical feature of the rover added, the GUI of the software has to update for the different features.

1. **Acceptance Test Plan**

**12.1 Displaying Camera Feed**

**12.1.1** The GUI will need to be able to display at least two video feeds simultaneously so that the driver can see the position and heading of the rover.

**12.1.2** A network connection will be established with the Rover from the Command PC and real time video feed will need to be streamed from the Rover PC to command.

**12.2 Sending Rover Commands**

**12.2.1** The Command PC will need to be able to issue commands to the Rover and have the Rover acknowledge the commands have been received on the Rover PC.

**12.3 Rover Navigation**

**12.3.1** The Command PC will need to be able to send a command to maneuver the

Rover in real time.

**12.3.2** The Rover commands will need to allow the Rover to move forward,

backward, turn left or right in place, and turn the front and back wheels.

**12.4 Articulating Arm Control**

**12.4.1** Command PC will need to be able to send commands to control the lateral movement of the Articulating Arm in real time

**12.4.2** The Command PC will need to be able to send commands to control the gripper of the Articulating Arm

**12.5 Communication**

12.5.1 The network communication will need to be able to send and receive

commands without and interruptions.

1. **Glossary**

**Command PC**: the PC which will be located locally and be used to view video and issue

commands to the rover

**Articulating Arm**: A robotic arm attached to the rover which will use to pick up and manipulate objects by the rover

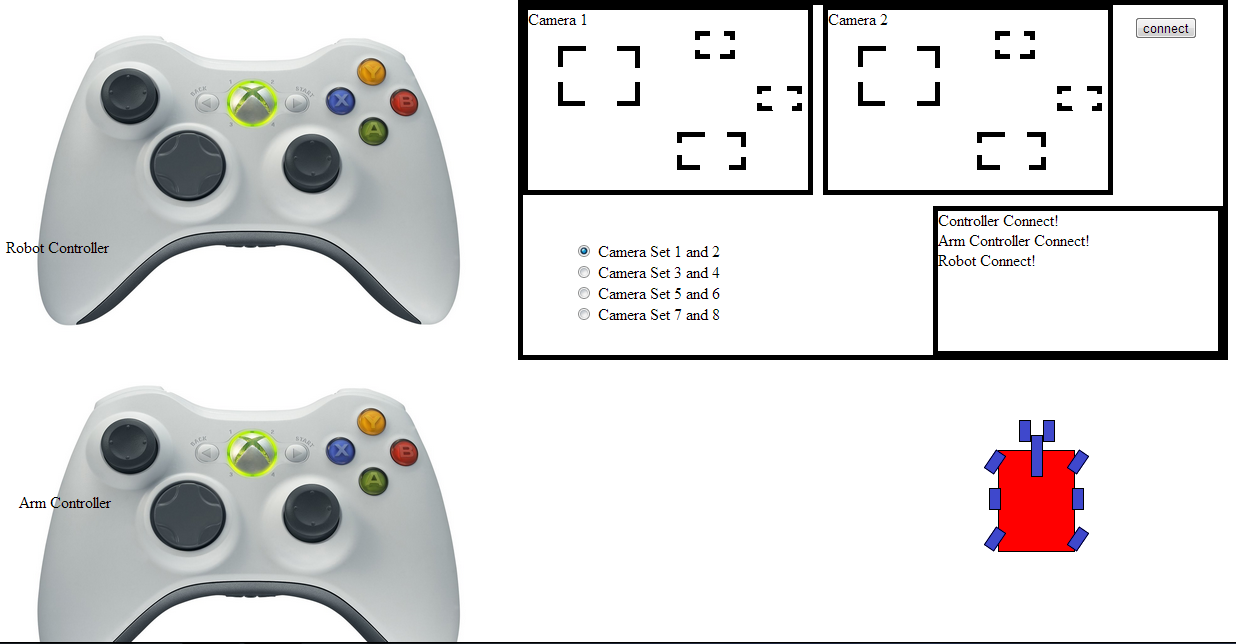
**GUI**: Graphical User Interface. The face of the application

**Rover PC**: the PC on the over that will serve as the basis for receiving commands as well as receiving information from the motors, sensors and cameras and sending that information back to the Command PC

**Java/C++/C#**: Object oriented programming languages

1. **Prototype Captures**

The following prototype is intended to give a basic understanding of how the Robot and video feeds will be controlled. The xbox Robot controller will be used for robotic movements while a GUI will be created to see the various video feeds. The xbox Arm controller will be used for robotic arm movements to pick up rocks.



*This is a basic idea of how the robot will be controlled and*

*the various video feeds viewed.*

1. **Reference Documents**

* http://nia-cms.nianet.org/RoboOps/index.aspx