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

The Rowan University Engineering team for the 2015 NASA Robo-Ops competition has requested assistance with the software aspects of their Mars Rover. RASC-AL Exploration Robo-Ops is an engineering competition sponsored by NASA and organized by the National Institute of Aerospace. This competition is open to both undergraduate and graduate students. 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.

This competition will be held at the NASA Johnson Space Center’s Rock Yard located in Texas. There will be 8 teams chosen to compete. Each team chosen to compete will receive a total of $10,000 to build their robot. The team is provided with the first half of the money after being accepted into the competition. The second half is given only after passing the mid project review. This review consists of a video conference with the steering committee. They are expecting a mobile prototype from the team to show all the progress that has been made. The prizes for this competition are a $6,000 first place prize, $4,000 for second place, and $2,000 for third place. There are also many recognition awards for various aspects of the competition. For example, there is an Olympic Matches award for completing/winning the most Olympic Matches in the various maps.

The NASA rock yard consists of 4 different maps for the rovers to compete on. These maps include the Rock Field, Lunar Craters, Sand Dunes, and Mars Hill. Each map is created to simulate other planets. There will be environmental obstacles such as hills, rocks, craters, etc. When a rock is found, it is to be collected via the articulating arm. Each rock is worth a certain amount of points. Each color represent a different point value. Generally small rocks are worth more points than bigger rocks. The goal is to collect as many points as possible to win the competition. The user, located at the Rowan Control Station in NJ, will view video feeds from the Rover remotely and be able to send commands to the Rover using an Xbox controller.

These commands allow the user to control the robots movements and views. The team located in NJ will have the ability to view one of the many cameras located on the Rover, and send movement commands accordingly. Video feeds from the cameras will also be streamed for the NASA steering committee to view during the competition. We have been asked to focus on the robotic controls, 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 qualities to be taken into consideration:

**Performance:** The Rover will require efficient means to transmit and receive data, since the competition will be live and users will need to operate in real time. This is also important in regards to the amount of video data that will need to be transmitted

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

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

Important risks to be taken into consideration:

**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.

**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.

**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.

1. **Application Context**

Currently, the Mars Rover is completely blind and immobile. There is no way to control any of its movements even in a close proximity. Our software will allow it to move and give sight to those moving it from a completely remote location.

The remote control software we are creating will let the users not only use the Rover if they are nearby, but allow them to control it from multiple states away. They will also now be able to view any of the cameras on board the Rover using the GUI. Users can toggle between the cameras, two at a time, to see what the Rover sees and move it appropriately.

1. **Functional Requirements**

**4.1 Video Feeds**

**4.1.1** A connection will be established between the Rover and the Command PC

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

**4.1.3** 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 arm.

**4.3 Communications**

**4.3.1** The rover’s onboard CPU must be able to connect with the main controllers

through a wireless connection.

**4.3.2** All the communications between the rover and controllers must be using 4G

network.

**4.3.3** Data must be able to transmit over long distances.

**4.3.4** The bandwidth of the 4G network must be able to handle multiple cameras

sending feedbacks and commands from the controllers.

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**

**Performance:** The Rover will require efficient means to transmit and receive data, since the competition will be live and users will need to operate in real time. This is also important in regards to the amount of video data that will need to be transmitted

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

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

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

**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.

**Robustness :** The rover 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 so that the engineering team and future viewers of the code can use it.

**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
* The customer must provide a basic prototype for testing out the basic functionality of the Rover.
* Rowan Engineers and Computer Science Majors will be able to use software created for the Rover.

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:** We must have basic working code that covers the basic features of the rover by October 22nd, 2014

**Phase 3:** We must have fully functioning code which covers all features and must be as efficient as possible as well as fully tested for correctness by November 26th, 2014.

1. **Potential Risks**

**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.

**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.

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

**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.

**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.

**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.

**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.

**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.

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

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

**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.

**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.

**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.

**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.

**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.

**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.

**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.

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

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

**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

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 Video Feeds**

**12.1.1** The Rover and Command PC will establish a connection to each other, signified by a message on the GUI and confirmed by the ability to manipulate the Rover

**12.1.2** The toggle button will cycle through all camera feeds not currently in use by the other display. Each display will toggle through 7 different angles, with the 8th being displayed constantly on the other.

**12.1.3** Both displays will show separate video feeds, and neither display will show the same feed as the other

**12.2 Robotic Controls**

**12.2.1** The move forward command will cause the Rover to move forward at various speeds depending on the position of the joystick.

**12.2.2** The move backward command will cause the Rover to move backwards at various speeds depending on the position of the joystick.

**12.2.3** While moving forward or backwards, the Rover is able to turn left or right. The turning angle depends on the position of the joystick.

**12.2.4** The Rover will turn left in place.

**12.2.5** The Rover will turn right in place.

**12.2.6** The articulating arm can be controlled via the controller.

**12.3 Communication**

**12.3.1** The Rovers onboard CPU is connected to the controller via the Command PC.

**12.3.2** The wireless connection established uses the 4G network.

**12.3.3** The connection established is able to transmit data over long distances.

**12.3.4** The 4G network is able to handle camera feeds and commands being sent back and fourth.

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

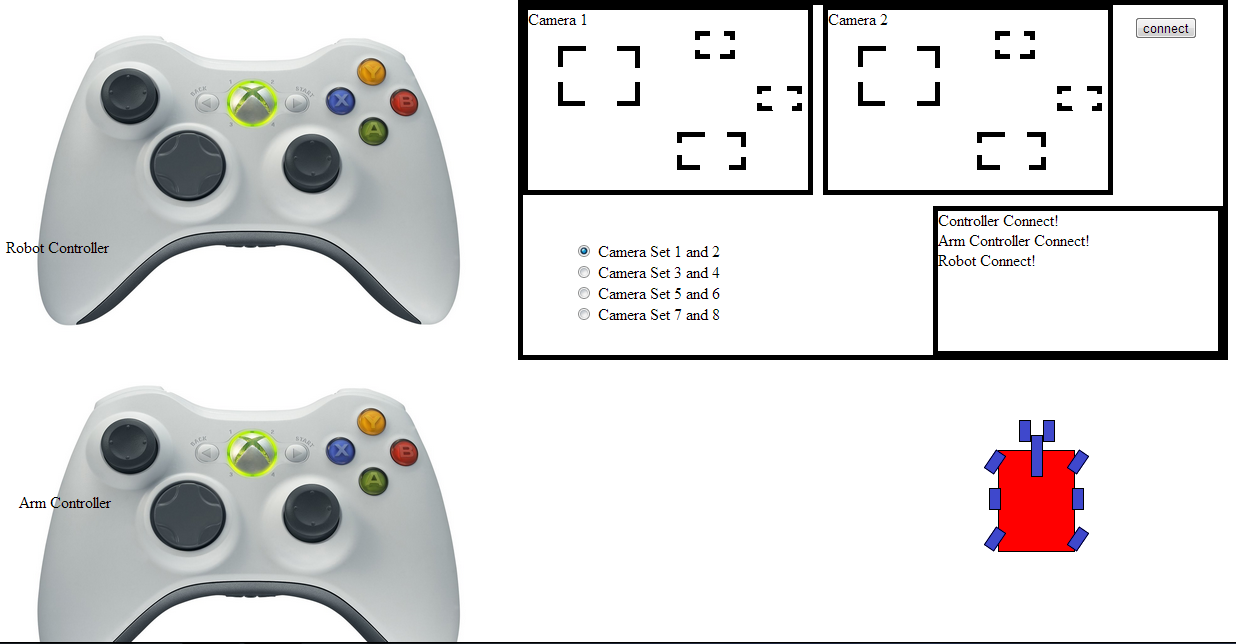
**GUI**: Graphical User Interface. The face of the application

**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 direction keys in Robot controller can control robot to different directions: forward, backward, left and right. When user turning the right analogue stick the different degrees, the front and back total four wheels will also turn into same degrees in robot. The xbox Arm controller will be used for robotic arm movements to pick up rocks. The forward and backward keys of Arm controller can control robot arm forward and back. The left and right keys can control robot arm turns to left and right. When left and right are continue click, the angle of the arm will increase/decrease, until it reaches to the its max free angular. When the analogue stick is clicked, the claw with be open/close. The GUI has two videos screens which can be changed when the GUI’s camera selection set changes. The system notification will notice users, when the controller gets powered on/off and the robot connect/disconnect with the system. Users can’t send orders to the robot unless the powers turned on and the GUI has been connected to the robot.



*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