**Detailed Design Documentation**

**Robo-Ops Competition**

**Controls and Communication**

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

**Software Engineering Fall 2014**

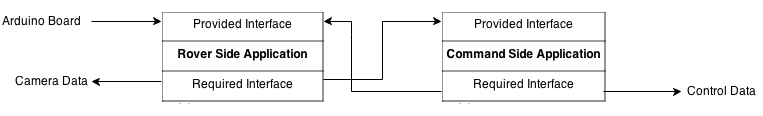
**Rowan University**

**Instructor: Dr. Adrian Rusu**

1. **Introduction**

This is the official design document for the NASA’s RASC-AL Robo-Ops planetary exploration rover competition rover. This document is based on the requirements explained by the SRS document. The system will allow the following requirements:

* Allow the user to wirelessly control the Rover while at the station located in NJ.
* Allow the user to see live camera feeds coming from the rover and switch between cameras at will.
* Allow the user to control the arm of the rover to pick up rocks as well as other basic controls such as move forward, backward and turn left and right.



**Figure 1.1 High-level architecture**

This design will allow the user to quickly send data wirelessly. A Graphical User Interface will also be implemented which will allow the user to easily control the cameras on the rover. A joystick will allow the user to easily control the rovers movements. These features will help the user control the rover and its respective cameras as quickly and fluidly as possible.

The rest of the document is organized in the following way: Chapter two contains detailed descriptions of each module found in the design, including the purpose, rationale and interfaces of each module. Chapter three contains a list of Abstract Data Types(ADT’s) used in the application. Chapter four will illustrate a COMPRISES diagram. Chapter five will illustrate a USES diagram, this will lay out the diagram in more detail. Lastly chapter six will describe a detailed test plan for the application.

1. **Module Design**

**2.1 Command Side Application Module**

**Purpose**

The purpose of this module is to provide the user with wireless control of the rover from afar. The module will allow the user to turn the rover, move the rover forward, backward, and control the arm of the rover. It will also allow the user to switch between camera feeds streamed live from the rover. This module also contains the GUI for the whole system, allowing the user to view information from the rover’s cameras.

**Rationale**

The rationale of this module is to give the user complete control of a rover from a distance in order to compete in the NASA’s RASC-AL Robo-Ops planetary exploration rover competition.

**The provided interface.**

The Command Side Application Module will not have a provided interface.

**The required interface**.

Define the required interface by listing all its

methods with descriptive names. The definition includes the names of

the methods, their parameters, and their return types. Provide a short

commentary describing each of the methods.

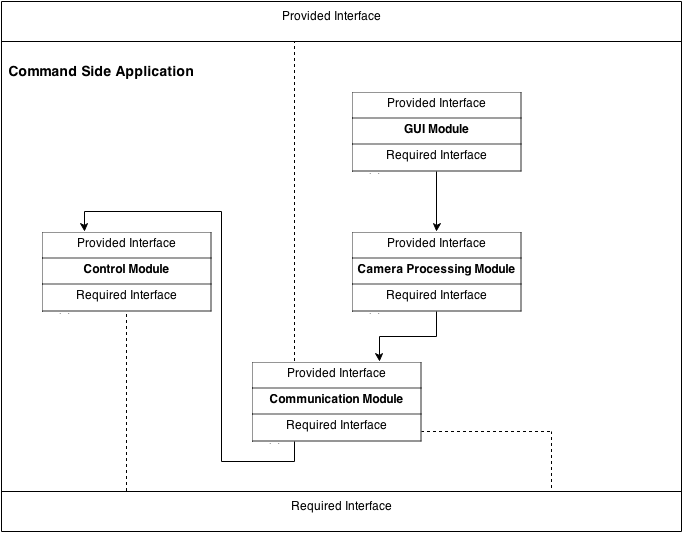
**Any other information.**

Add any other information that may aid in the

understanding of the module, such as, for example, its inheritance

hierarchy or performance constraints. Only include this part if

necessary.



**Figure 2.1 High-level Design for Command Side Application Module**

**2.2 Controller Module**

**Purpose**

The purpose of this module is provide the connection controllers and the local PC. So that when the users create actions on controllers, this module can figure out which controller is sending actions, and send the signal to the local PC, then to the Robot PC, Finally to the Robot’s motors and arm. This module can handle signals and send to connection module.

**Rationale**

Without this module, the PC wouldn’t able to send orders to the rover. Rover will not able to function for action.

**2.2.1 Motor Control Module**

**Purpose**

The purpose of this module is receive inputs from motor joystick, specify actions for the robot and send actions to the robot motors, so that the motors can move forward, backward and turns.

**Rationale**

Without this module users can’t create motor actions to the robot. Without motor actions, robot will not able to move.

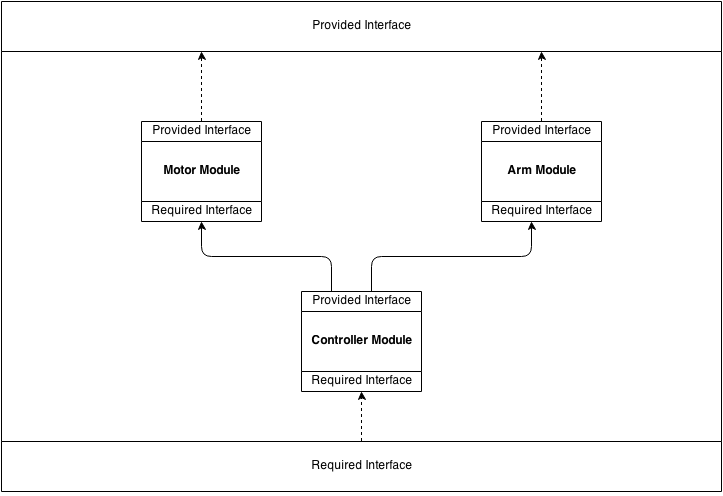
**2.2.2 Articulating Arm Control Module**

**Purpose**

The purpose of this module is receive inputs from arm joystick, specify actions for the robot and send actions to the robot arm, so that the arm can move base on its degree of freedoms.

**Rationale**

Without this module users can’t create arm actions to the robot. Without arm actions, robot will not able to move the arm and will not able to pick up objects.



**2.3 Camera Feed Control Module**

**Purpose**

The purpose of the cameras is to allow the user to see the rover’s environment and choose the most optimal path for the rover. The user will be allowed to switch between the camera feeds. This module will function as a listener which will be attached to the visualization that will update the cameras.

**Rationale**

The rationale of this module is to allow the user to see where the rover is going in order to plot the best path to take.

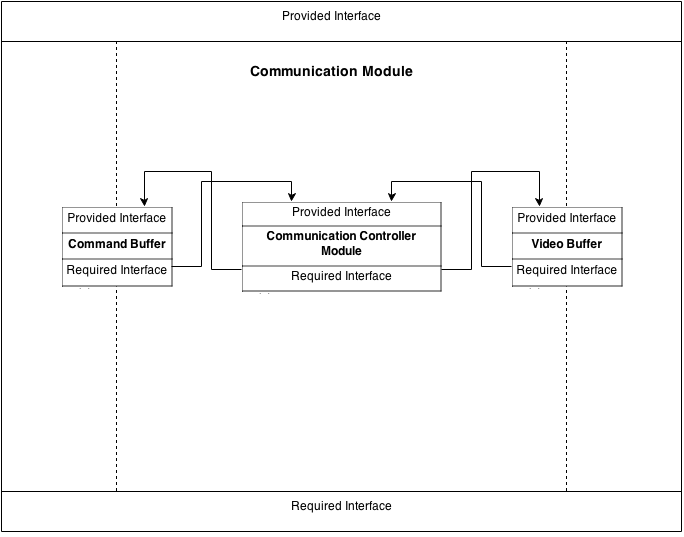
**2.4 Communications Module**

**Purpose**

The purpose of this module is to manage sending commands and receiving video feed backs over network connections between the rover’s onboard PC and the command PC. There is also a website hosted on Rowan’s network to act as the middle man to handle receiving video feeds and taking commands from the command PC then send it over to the rover’s onboard PC.

**Rationale**

This module was created in order to enable the Mission Control and the Rover to be able to communicate over a long distance. Also it eases out the onboard PC’s computation by handling the video streaming.



**Figure 2.4 High Level Architecture of Communication Module**

**Provided interface**

**void sendCommands(byte command)**

Description: This function sends out commands to the Rover’s onboard PC

Parameters: byte command

Returns: none

Throws: none

**Required interface**

**2.5 Rover Onboard Module (Dan)**

**Purpose**

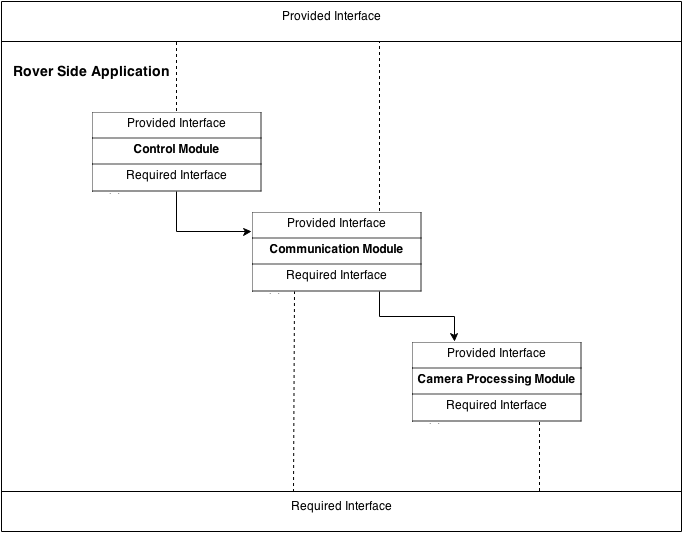
The purpose of the rover onboard module is to manage the flow of information between the command PC and rover motors, arm and camera. A buffer holds commands issued from the command PC. The commands are continually processed from the buffer and converted into a format that the Arduino systems can used to control the command motors. The video sent from the camera will be processed by the onboard rover PC so it can be sent back to the communications module in an efficient and useable format. The rover will also be able to receive commands to only return certain sets of the video feeds.

**Rationale**

This module is necessary to serve as an interlink between the command PC and the motors so that there is a way to control and command the rover remotely.

**High Level Module Design**

The module is broken down as shown below.



**Figure 2.5 High-level Architecture of Rover Side Application Module**

**2.5.1 Rover Buffer Module**

**Purpose**

The purpose of the buffer module is to hold command information as it is

received across the network from the Communications module. It will hold this information until the Rover Command Processing module is ready to process it.

**Rationale**

The rover will not be able to immediately process any command as it is sent

across the network so a buffer is needed to hold a queue of commands to be processed in order as the rover command processing module handles them.

**Provided Interface**

**boolean isReady()**

Definition: Checks to see if the buffer is ready to receive information.

Parameters: None

Returns: A boolean representing whether or not the information is ready.

Error: IOException

**BufferUnit processBuffer()**

Definition: Processes some bytes of information that have been received from

the communications module. These bytes should include the command itself and the intended target of the command whether it be motor or camera control. It will convert these bytes in a BufferUnit that will be sent to the rover command module for processing.

Parameters: None

Returns: A BufferUnit which holds bytes of command info and a destination

motor.

Error: EndOfBufferException

**Required Interface**

**2.5.2 Rover Command Processing Module**

**Purpose**

The purpose of the rover command process module is to take commands that

have been received in the rover buffer, process them into a Arduino useable format, and send them to the Arduino motor controls.

**Rationale**

A module is needed to coordinate handling of all the different commands that are

received in the buffer. The commands need to be sent to the proper place in usable format.

**Provided Interface**

**void sendCommandToMotor(byte commandValue, int motorNumber)**

Defintion: Sends a command value that the rover has received from the

command PC on to a specific motor as identified by motor a motor number.

Parameters: A byte, representing a value between 0-255 for how far the motor

needs to turn

An int, representing the motor which the command is ultimately intended for

Returns: none

Errors: IndexOutOfBoundsException

**void processBufferUnit(BufferUnit command)**

Definition: Converts the command that has been received over the network

into a command that can be used by the Arduino.

Parameter: A BufferUnit, representing command information and intended

destination that was being held in the Buffer module.

Returns: none

Errors:

**Required Interface**

**Rover Buffer Module**

BufferUnit processBuffer()

**2.5.3 Rover Camera Processing Module**

**Purpose**

The purpose of this module is to take the encoded video file received from the

camera and prepare it to be sent to the command PC.

**Rationale**

This module is needed in order to make sure that the encoded video is in a

format that can actually be sent through the rover communications module. Otherwise the video will not be able to be sent back to the command PC.

**Provided Interface**

**transferableVideo prepareVideo(encodedVideo videoFeed)**

Definition: Takes encoded video receieved from the camera and prepares it

for transmission across the internet to the command PC.

Paramter: encodedVideo, a representation of the format that the camera video

is being sent in

Returns: transferableVideo, a representation of the format that can be

transmitted across the network to the command PC.

Error:

**Required Interface**

**2.5.4 Rover Outbound Communication Module**

**Purpose**

The purpose of this module is to send processed video feed across the internet

back to the command PC.

**Rationale**

This is needed so that the GUI can receive the real time video feed from the

cameras.

**Provided Interface**

**boolean openConnection()**

Definition: Opens up an outbound connection with the command PC

Parameter: None

Returns: A boolean, representing whether or not a connection was

successfully opened with the command PC.

Error: IOException

**void sendVideoFeeds(Video videoFeeds)**

Definition: sends out video feeds to the website

Parameter: Video videoFeeds

Returns: None

Throws: None

**boolean receivedCommand()**

Description: This function will returns the boolean value of whether it receives the command from the command PC.

Parameters: none

Returns: boolean

Throws: none

**Required Interface**

**Rover Camera Processing Module**

transferableVideo prepareVideo(encodedVideo videoFeed)

1. **Abstract Data Types**

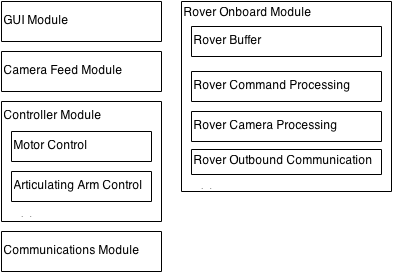
BufferUnit {

ArrayList<byte> commands

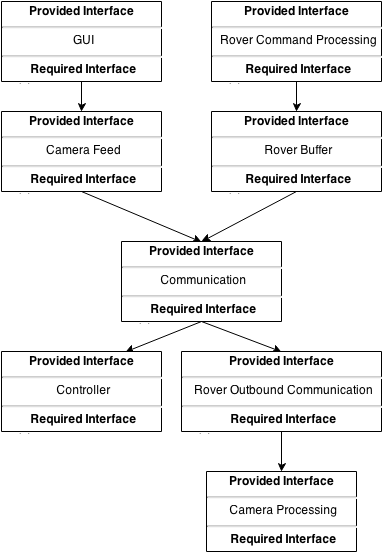
int intendedDestination

}

1. **COMPRISES Diagram**



1. **USES Diagram**



1. **Integration Test Plan**

**Approach**

Our team will be using a bottom up approach. More testing will be done as more

motors and technology become available to us. We will start tests to make sure

our joystick works. Then we will move on to sending commands to another

computer. Eventually making our way to sending commands back and fourth

between the Rover on-board CPU.

**Order of Integration**

Below are the various test phases that will be performed as more technology for

the Rover becomes available to us.

|  |  |  |
| --- | --- | --- |
| **Integration Group** | **Driver(s) Needed** | **Stub(s) Needed** |

**Phase 1: Joystick Controls**

|  |  |  |
| --- | --- | --- |
| Move Forward | Joystick Driver | None |
| Move Backward | Joystick Driver | None |
| Move Left | Joystick Driver | None |
| Move Right | Joystick Driver | None |
| Set Speed | Joystick Driver | None |
| Change Camera | Joystick Driver | None |
| Move Arm | Joystick Driver | None |

**Phase 2: Communications Modules**

|  |  |  |
| --- | --- | --- |
| Send Signal | Communications Driver | Signal Stub |
| Receive Signal | Communications Driver | None |

**Phase 3: Motor Control**

|  |  |  |
| --- | --- | --- |
| Motor A Movement | Motor Driver | None |
| Motor A Rotation | Motor Driver | None |
| Motor B Movement | Motor Driver | None |
| Motor B Rotation | Motor Driver | None |
| Motor C Movement | Motor Driver | None |
| Motor D Movement | Motor Driver | None |
| Motor E Movement | Motor Driver | None |
| Motor E Rotation | Motor Driver | None |
| Motor F Movement | Motor Driver | None |
| Motor F Rotation | Motor Driver | None |
| Arm Movement | Motor Driver | None |
| Arm Rotation | Motor Driver | None |

**Phase 4: Video Feeds**

|  |  |  |
| --- | --- | --- |
| Send Video | Video Driver | Video Stub |
| Receive Video | Video Driver | None |

**Driver Definitions:**

**Joystick Driver:** This driver tests the functionality of the joystick. It will allow commands to be given from the joystick and visually display if they are being received by the program.

**Communications Driver:** This driver tests the functionality of our communications. It will create signals to be packaged and sent/received between the Command Station and the Rover.

**Motor Driver:** This driver tests the functionality of our Motors. It will provide necessary signals to cause each motor to move or rotate.

**Video Driver:** This driver tests the functionality of the video streaming capabilities.

**Test Cases:**

**Phase 1: Joystick Controls**

**ID:** Move Forward

**Description:** Test ability to send Move Forward signal to computer.

**Data Input:** None

**Expected Output:** Notification that the correct signal was successfully received.

**Environmental Constraints:** Properly functioning joystick.

**Testing Constraints:** None

**ID:** Move Backward

**Description:** Test ability to send Move Backward signal to computer.

**Data Input:** None

**Expected Output:** Notification that the correct signal was successfully received.

**Environmental Constraints:** Properly functioning joystick.

**Testing Constraints:** None

**ID:** Move Left

**Description:** Test ability to send Move Left signal to computer.

**Data Input:** None

**Expected Output:** Notification that the correct signal was successfully received.

**Environmental Constraints:** Properly functioning joystick.

**Testing Constraints:** None

**ID:** Move Right

**Description:** Test ability to send Move Right signal to computer.

**Data Input:** None

**Expected Output:** Notification that the correct signal was successfully received.

**Environmental Constraints:** Properly functioning joystick.

**Testing Constraints:** None

**ID:** Set Speed

**Description:** Test ability to send Set Speed signal to computer.

**Data Input:** None

**Expected Output:** Notification that the correct signal was successfully received.

**Environmental Constraints:** Properly functioning joystick.

**Testing Constraints:** None

**ID:** Change Camera

**Description:** Test ability to send Change Camera signal to computer.

**Data Input:** None

**Expected Output:** Notification that the correct signal was successfully received.

**Environmental Constraints:** Properly functioning joystick.

**Testing Constraints:** None

**ID:** Move Arm

**Description:** Test ability to send Move Arm signal to computer.

**Data Input:** None

**Expected Output:** Notification that the correct signal was successfully received.

**Environmental Constraints:** Properly functioning joystick.

**Testing Constraints:** None

**Phase 2: Communications Module**

**ID:** Send Signal

**Description:** Test the ability to package and send a signal to a designated IP.

**Data Input:** A signal to mimic those inputted from the joystick.

**Expected Output:** Notification that the signal was successfully sent.

**Environmental Constraints:** Network is functioning properly.

**Testing Constraints:** None

**ID:** Receive Signal

**Description:** Test the ability to unpack and read signal.

**Data Input:** Packaged signal to mimic format of sent signal.

**Expected Output:** Notification that the correct signal was received successfully.

**Environmental Constraints:** Network is functioning properly.

**Testing Constraints:** None

**Phase 3: Motor Control**

**ID:** Motor A Movement

**Description:** Test out the ability of Motor A

**Data Input:** Move forward command

**Expected Output:** Motor will move forward

**Environmental Constraints:** None

**Testing Constraints:** We don’t know when we are actually going to get a

prototype

**ID:** Motor A Rotation

**Description:** Test the rotation ability of the Motor

**Data Input:** Number of rotations

**Expected Output:** Actual number of rotations on motor A

**Environmental Constraints:** None

**Testing Constraints:** We don’t know when we are actually going to get a

prototype

**ID:** Motor B Movement

**Description:** Test out the ability of Motor B

**Data Input:** Move forward command

**Expected Output:** Motor will move forward

**Environmental Constraints:** None

**Testing Constraints:** We don’t know when we are actually going to get a

prototype

**ID:** Motor B Rotation

**Description:** Test the rotation ability of the Motor

**Data Input:** Number of rotations

**Expected Output:** Actual number of rotations on motor B

**Environmental Constraints:** None

**Testing Constraints:** We don’t know when we are actually going to get a

prototype

**ID:** Motor C Movement

**Description:** Test out the ability of Motor C

**Data Input:** Move forward command

**Expected Output:** Motor will move forward

**Environmental Constraints:** None

**Testing Constraints:** We don’t know when we are actually going to get a

prototype

**ID:** Motor D Movement

**Description:** Test out the ability of Motor D

**Data Input:** Move forward command

**Expected Output:** Motor will move forward

**Environmental Constraints:** None

**Testing Constraints:** We don’t know when we are actually going to get a

prototype

**ID:** Motor E Movement

**Description:** Test out the ability of Motor E

**Data Input:** Move forward command

**Expected Output:** Motor will move forward

**Environmental Constraints:** None

**Testing Constraints:** We don’t know when we are actually going to get a

prototype

**ID:** Motor E Rotation

**Description:** Test the rotation ability of the Motor

**Data Input:** Number of rotations

**Expected Output:** Actual number of rotations on motor E

**Environmental Constraints:** None

**Testing Constraints:** We don’t know when we are actually going to get a

prototype

**ID:** Motor F Movement

**Description:** Test out the ability of Motor F

**Data Input:** Move forward command

**Expected Output:** Motor will move forward

**Environmental Constraints:** None

**Testing Constraints:** We don’t know when we are actually going to get a

prototype

**ID:** Motor F Rotation

**Description:** Test the rotation ability of the Motor

**Data Input:** Number of rotations

**Expected Output:** Actual number of rotations on motor F

**Environmental Constraints:** None

**Testing Constraints:** We don’t know when we are actually going to get a

prototype

**ID:** Arm Movement

**Description:** Test the ability to move the Articulating Arm.

**Data Input:** Degrees of movement.

**Expected Output:** Arm to move as expected.

**Environmental Constraints:** None

**Testing Constraints:** Prototype not yet received.

**ID:** Arm Rotation

**Description:** Test the ability to rotate the Articulating Arm.

**Data Input:** Degrees of rotation.

**Expected Output:** Arm to rotate as expected.

**Environmental Constraints:** None

**Testing Constraints:** Prototype not yet received.

**Phase 4: Video Feeds**

**ID:** Send Video

**Description:** Video feed is packed and sent out to a specific location.

**Data Input:** Video feed

**Expected Output:** Notification that the feed was sent successfully.

**Environmental Constraints:** Proper 4G network connection.

**Testing Constraints:** None

**ID:** Receive Video

**Description:** Video feed is received and viewable.

**Data Input:** Packaged video

**Expected Output:** Display video

**Environmental Constraints:** Successful 4G network connection

**Testing Constraints:** None