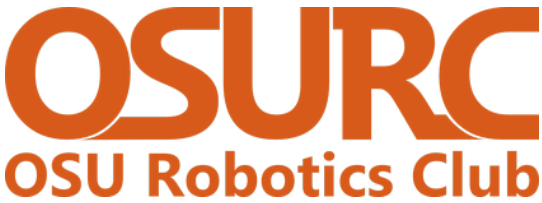

CS CAPSTONE PROBLEM STATEMENT

OCTOBER 10, 2017

OSU ROBOTICS CLUB MARS ROVER GROUND STATION

PREPARED FOR

OSU ROBOTICS CLUB

NICK MCCOMB

PREPARED BY

GROUP 30

KENNETH STEINFELDT
CHRISTOPHER PHAM
CORWIN PERREN

Abstract

This project involves the creation of the user interface and associated back-end systems needed to control the OSU Robotics Club's Mars Rover. The club's Rover implements numerous control and sensor systems, all of which require ground station software to facilitate quick and easy use of the Rover during timed competitions. The ground station software will send the Rover control information received via Joystick input or GUI interaction, as well as process and interpret status data and multiple, concurrent video streams. An integrated mapping system will allow the Rover to run autonomously using predefined way-points. More complex readouts will dynamically show critical system components, such as a live view of arm joint positions and compass showing current heading and markers for way-points and points of interest.

Contents

1 Problem Description

The OSU Robotics Club Mars Rover Ground Station is the primary point of contact between the operating user and the Rover itself. In order for the Mars Rover team to be successful during competition the software must be easy to use and free from crashes and major errors. Users will control the Rover through a combination of two USB joysticks as well as a keyboard and mouse, across two 24" 1080p monitors. With this interface, users will need the ability to see and record the one or more video streams being sent back from the Rover as well as the GPS position of the Rover on a map of the competition area.

Readouts on this software will need to display status information such as accessory connection states, GPS heading and speed, drive motor power, battery and system voltages, network latency, radio signal strength, and science data. USB joysticks will allow users to drive the Rover as well as control secondary systems such as the arm and main-navigation camera gimbals. Due to the competition environment this software will need to be able to handle potential network dropouts, large spikes in network latency, or frequent packet loss. As testing of the Rover itself is the single largest priority, rapid prototyping and development of this software is desired. Additionally, the ground station software must be written keeping in mind that future Rover software teams will likely reuse and modify the code for future iterations of the robot. Ideally, even incomplete versions of the software will need to be available to test specific components of the Rover during its assembly phase. When properly functioning, this software will need to allow a trained user to effectively and efficiently use the Rover to complete competition tasks.

2 Proposed Solution

In order to meet the goals of this project, the team's proposed solution will be to design the ground station software using Python 3, the application framework "QT" through use of the PyQt5 library, and a yet to be determined version of Robot Operating System (ROS). Since rapid prototyping will be needed, Python will allow our team to more quickly develop and test prototypes while also making it easier for future members of Rover software teams to understand what was written. The use of the QT framework will greatly simplify the creation of a sleek user interface and allow the team to make quick and easy adjustments to the layout in the future. By using ROS, we will also be able to accelerate the design time by using the ROS topic subscription and broadcasting services built into the framework. This feature in particular will let us avoid creating custom command control packets that would be hard to reuse without major modifications for future years, as well as allow the Rover to natively accept these commands without the need for an interpreter node.

3 Performance Metrics

The primary metric for whether the software has met the criteria for the project will be if the team president, a team officer, and at least three other Rover team members personally use the ground station software to control the rover and sign-off that it is adequately responsive, intuitive to use, and does not crash even under unideal conditions such as partial message loss or high latency. For these tests, unideal conditions will be simulated via purposeful improper placement of Rover radios. Additionally, there will be empirical measurement of software traits. Examples off these measurements include video stream frame rates, maximum latency values for GUI element updates, and maximum latency for control input to rover response.