CS CAPSTONE TECHNOLOGY REVIEW

NOVEMBER 21, 2017

OSU ROBOTICS CLUB MARS ROVER GROUND STATION

PREPARED FOR

OSU ROBOTICS CLUB

NICK MCCOMB

PREPARED BY

CORWIN PERREN

Abstract

This document will cover my personal team role in developing the OSU Robotics Club's Mars Rover Ground station software. Additionally, it will cover different options for robotics frameworks, radio systems, and joysticks that the Rover team and specifically the Ground Station software team may end up using to complete our software package.

Contents

1 Personal Role 2

2 Goals 2

3 Robotics Frameworks 2

3.1 Overview . . . 2

3.2 Criteria . . . 2

3.3 Potential Options . . . 3

3.3.1 Custom Framework . . . 3

3.3.2 Mobile Robotics Programming Toolkit (MRPT) . . . 3

3.3.3 Robot Operating System (ROS) . . . 3

3.4 Discussion . . . 4

3.5 Conclusion . . . 4

4 Radio System 4

4.1 Overview . . . 4

4.2 Criteria . . . 5

4.3 Potential Options . . . 5

4.3.1 RockBLOCK Mk2 Iridium SatComm . . . 5

4.3.2 LoRa 433MHz Radio . . . 5

4.3.3 Ubiquity Rocket M2 . . . 5

4.4 Discussion . . . 6

4.5 Conclusion . . . 6

5 Joysticks 6

5.1 Overview . . . 6

5.2 Criteria . . . 6

5.3 Potential Options . . . 7

5.3.1 Logitech Saitek X52 . . . 7

5.3.2 Logitech Saitek Evo . . . 7

5.3.3 Hercules Thrustmaster T16000M . . . 7

5.4 Discussion . . . 7

5.5 Conclusion . . . 8

6 References 8

1 Personal Role

My role on the Mars Rover Ground Station team will primarily be as a programmer writing code to interface with the team's Mars Rover. As part of this interfacing, the software I help write will allow for remote control of the Rover, the showing of video streams broadcast from the Rover, and the showing of Rover status and mapping information. An additional role I will have will be as a bridge between the rest of our capstone team and the Mars Rover group, as needed. In the past, I have both been a member of the Mars Rover team and the electrical team lead, so my knowledge of the competition as well as those in charge of the Rover team will help facilitate easy communication.

2 Goals

The goals of the rest of this document are to cover three separate technological aspects of the Mars Rover Ground Station that have multiple viable solutions. Evaluation of three of these options will then culminate in choosing one of one to be used on the Ground Station. The first technology to be looked at will be the robotics framework the software will be interfacing with. As the Mars Rover is a complicated robot, choosing the right option here will affect how difficult it will be to interface with the Rover's systems. The second technology will cover the radio system options the Ground Station could use to communicate with the Rover. Selection of this system will determine how much and how fast data can be transmitted between the Ground Station and Rover, both of which are major factors in how usable the Rover is during competition. The final technology will be the options for USB joysticks that will be used to drive the Rover remotely using this Ground Station software.

3 Robotics Frameworks

3.1 Overview

When working with large robotic systems there are often many sensors, actuators, computers, and cameras that all need to work in unison to make the platform perform its desired task. In order to facilitate this intercommunication most robots use off the shelf robotics frameworks so that programmers can focus on implementing higher level features. The result is that less time is spent on lower leveling coding for things like sending messages between robot system nodes and more time can be spend implementing features such as obstacle avoidance. As the Mars Rover Ground Station software is interfacing with a complicated robot, it is definitely important to consider different options for these frameworks to choose the one best suited to the project.

3.2 Criteria

- *Feature Coverage*: Ideally the framework should take care of as many lower level tasks as possible including, but not limited to, inter-node communications, simplified computer vision, localization, motion-planning, and built-in simulators.
- *Hardware Compatibility*: The framework would preferably have pre-built nodes/libraries/drivers for common hardware like GPS and cameras.
- *Documentation*: In order to facilitate rapid development, ample and well-maintained documentation is a must.
- *Community / Resource Availability*: To supplement official documentation, the framework should have a well-developed community to allow for easy answering of framework programming questions when encountered as well as example code when needed.

- *Development Time*: Due to the limited time constraints, the framework used should allow for fast development of the software. This component will take into account all the previous criteria.

3.3 Potential Options

3.3.1 Custom Framework

This option involves writing a framework from scratch in-house. Due to its popularity within the Mars Rover team, Python would most likely be used as the language of choice to develop such a system. A custom framework would have to incorporate writing nodes for handling GPS localization, camera processing, computer-vision based obstacle avoidance, drive system control, arm system control, and many others with little to no starting code. And important aspect of this system, with respect to the ground control software, is that a custom system would need specialty sending and receiving nodes for handling transmission of control, status, and video data.

- *Feature Coverage*: Complete coverage is possible.
- *Hardware Compatibility*: Complete hardware support is possible.
- *Documentation*: Only pre-made documentation for libraries is available.
- *Community / Resource Availability*: Reliant on availability for any libraries used.
- *Development Time*: Development will take an incredibly long amount of time.

3.3.2 Mobile Robotics Programming Toolkit (MRPT)

The Mobile Robotics Programming Toolkit is a robotics framework developed to handle the difficult aspects of robot localization, mapping, computer vision, and motion planning. While this does not cover all of the modules needed for the Mars Rover team, nor the ground station, it would greatly simplify some of the design work for the more challenging aspects of the Rover. Custom nodes would still have to be made for drive and arm system control as well as custom messaging nodes.

- *Feature Coverage*: Partial coverage. Still requires writing complex nodes for some features.
- *Hardware Compatibility*: Partial hardware support. Custom additions would be needed to support the rest of the robot hardware.
- *Documentation*: A reasonable number of official tutorials are available.
- *Community / Resource Availability*: Help can be found on stackoverflow.com with the search tag ”mrpt”.
- *Development Time*: Development will take a moderate amount of time. This option still requires a significant amount of custom node development.

3.3.3 Robot Operating System (ROS)

ROS is the most popular robotics framework in the world, and it shows in terms of the complete package it provides. Nodes can be found, both official and unofficial, for most features the Robotics Club’s Mars Rover might need. This includes localization and mapping, computer-vision, obstacle avoidance, motion planning, arm and drive system nodes, and and robust messaging framework. For any nodes that are not complete, they often provide a great starting point so we would at least not be starting from scratch.

- *Feature Coverage*: Excellent coverage. Will require some modification of some feature nodes, but most are pre-built and ready to use.

- *Hardware Compatibility*: Excellent hardware support. ROS comes with support for many robotic hardware tools, and abstracted nodes also for easy modification for custom hardware when needed.
- *Documentation*: Excellent documentation and tutorials on ROS's website.
- *Community / Resource Availability*: Many forums and sites provide help with ROS, as well as many Youtube video series dedicated to ROS.
- *Development Time*: Development time will be greatly reduced due to pre-built nodes and great documentation.

3.4 Discussion

In terms of feature coverage, ROS seems to be most complete. However, depending on how difficult implementing the remaining nodes might be, MRPT might not be too far off in terms of complexity. A custom solution has the benefit of having complete coverage once the code is written, but would take considerably longer to write overall.

With regards to hardware compatibility, it's a similar spread as with feature coverage. ROS is most compatible with the least amount of effort from the start. MRPT would require a fair bit of work to get working with a lot of the hardware that is not directly related to mapping, localization, and computer-vision. The custom solution again has the benefit of having complete hardware support once done, but could require a very large amount of time to get all hardware working.

Documentation-wise, the custom solution is the worst option. Any development will depend on general programming documentation for the language used, and any that exists for libraries in that language. MRPT has some documentation, but it's mostly limited to tutorials and provides less in-depth information for programmers wanting to adjust MRPT to a custom application. ROS, on the other hand, has excellent documentation and many tutorials.

The community and resource availability for the custom solution is essentially non-existent. Any community and resources would have to be internal to the Rover team or Ground Station software team. MRPT at least has a small community and is active on stackoverflow to answer questions when needed. ROS's community and availability is excellent by contrast. There are a multitude of forums, but official and unofficial providing support, and ample third party resources providing examples and videos.

Lastly, when it comes to development time ROS appears to win in this category. The combination of many pre-built nodes plus excellent documentation and community support helps make development much speedier than other options. MRPT is next best, with at least a starting framework to begin with and build off of. The custom solution would be a nightmare in terms of development time and would likely not be complete before competition.

3.5 Conclusion

For the choice of a robotics framework ROS definitely seem to be most fully-featured and quickest to implement framework and wins out in every single category compared to MRPT or a custom solution. Additionally, the fact that ROS has been actively developed for ten years shows in its mature and robust nodes and documentation that should allow the Rover team and our Ground Station software team to more quickly develop the desired application.

4 Radio System

4.1 Overview

During the University Rover Challenge competition, most of the events that take place are remote control events where the user must remotely operate their Rover from a remote base station. During both the remotely operated and autonomous events, data will need to be sent back and forth between the Rover and its base station. As such, the radios used make a huge difference in how well the Rover performs. Here, we'll look into a few different options for radio systems.

4.2 Criteria

- *Cost*: The radio must be as low cost as possible while still providing all needed functions.
- *URC Radio Rules*: The radio must meet the URC competition radio requirements.
- *Data Throughput*: The radio must allow for high enough data throughput to allow for comfortable remote driving of the Rover.
- *Range*: The radio must be able to reach at least 1km away per URC requirements.

4.3 Potential Options

4.3.1 RockBLOCK Mk2 Iridium SatComm

This satellite modem would allow for remote communications with the Rover anywhere the Rover has line of sight to the open sky. While more difficult to come by, and more difficult to interface with, these radios would near guarantee consistent communication between the Rover and ground station due to the open nature of the Utah desert where the competition will take place.

- *Cost*: The radio is expensive at $250, and also has costs per message.
- *URC Radio Rules*: This radio would meet URC requirements with prior approval.
- *Data Throughput*: The data throughput of this radio would be abysmal at one message every 10 seconds.
- *Range*: Range is essentially unlimited in the open desert of the competition area.

4.3.2 LoRa 433MHz Radio

These small serial radios are cheap and easy to use, but have low power output and require an amateur radio license to operate. These radios are quite easy to acquire and provide a simple abstract serial-over-radio interface that is easy to use.

- *Cost*: The radio is incredibly cheap at $20 each.
- *URC Radio Rules*: This radio would meet URC requirements with prior approval and licensing.
- *Data Throughput*: The data throughput of this radio would be quite slow at 19.2 kbaud per second, but enough to drive the Rover and receive status messages.
- *Range*: Line of sight these radios are rated at 2km.

4.3.3 Ubiquity Rocket M2

The Rocket M2 radios are 2.4GHz long-range WiFi radios that provide a remote Ethernet link between two stations. These radios are reasonably easy to come by, and while requiring some networking knowlege to set up, provides a simple-to-use interface that natively will integrate with Robotics Frameworks such as ROS.

- *Cost*: The radio is expensive at $250 each.
- *URC Radio Rules*: This radio meets URC requirements as it is on the pre-approved radios list.

- *Data Throughput*: The data throughput of these radios are up to 100MBit per second, enough to do control and status data as well as Rover video.
- *Range*: Line of sight, these radios have been tested at up to 1.25km by Rover team members.

4.4 Discussion

From a cost perspective, the LoRa radios definitely win out at a mere $20. The Rocket M2 and RockBLOCK are matched in price, but barring being beyond the range limit of the M2 radios, the RockBLOCK is not a very good contender in terms of throughput for price.

All of these radios are valid by URC competition rules, although both the satellite radio and LoRa radio would require express prior permission from the URC judges.

In terms of data throughput, the Rocket M2 is an easy winner, especially when considering the fact that it's fast enough to also send video data over rather than having to have a second radio for video transmission. The LoRa radio would be reasonable if only control and status information was being sent, but the satellite RockBLOCK would basically be useless for remote operation. To use the RockBLOCK you'd have to send the Rover way-points and let the Rover drive on its own.

Range-wise the satellite modem is the best option, but overkill for what the requirements of the competition are. LoRa is the next best option with 2km line of sight, but it is worth noting that the M2 radios also have a range greater than what is needed during competition.

4.5 Conclusion

Not surprisingly the RockBLOCK is not a reasonable option for this project. However, both the LoRa and M2 radios have enough data throughput and range to make it through the competition happily. With this in mind, the Rocket M2 radios are the optimal radio to use. While they are expensive at $250, they have enough bandwidth to transmit video data as well as control data meaning the team can save the costs of a secondary radio system for video.

5 Joysticks

5.1 Overview

The ground station software's primary function is to allow for remote control of the Rover via the use of joysticks. Therefore, it is not surprising that the choice of joystick is important as it is the primary point of contact between a driver and the Rover itself. Below we'll compare a few different choices for USB joysticks.

5.2 Criteria

- *Cost*: Due to cost restrictions imposed by the competition rules, keeping the cost as low as possible is a priority.
- *Control Feature Set*: The joystick chosen should have pitch, roll, a four position hat stick, at least five buttons, and a throttle slider.
- *Ambidextrous Grip*: As two joysticks will be bought, one to use with the left hand and one with the right, it must be one capable of being used with either hand from the factory.
- *Reviews*: User reviews for the joystick must imply that the joystick is well-made and is unlikely to fail during competition.

5.3 Potential Options

5.3.1 Logitech Saitek X52

The Saitek X52 Flight Control joystick has a large airplane style throttle control as well as a normal ambidextrous joystick. The throttle unit and joystick are separate pieces that are connected via a cable. [1]

- *Cost*: Expensive at $142.
- *Control Feature Set*: Exceeds minimum feature set.
- *Ambidextrous Grip*: Yes
- *Reviews*: This joystick has slightly better than average reviews, but complaints seem to focus on build quality issues where the joystick wears out prematurely.

5.3.2 Logitech Saitek Evo

The Saitek Evo joystick is a simple single-piece unit that has nine buttons, an 8-way POV hat, and a single throttle control. This is a pretty generic joystick. [2]

- *Cost*: Very cheap at $45.
- *Control Feature Set*: Exceeds minimum feature set.
- *Ambidextrous Grip*: Yes
- *Reviews*: This joystick gets average reviews, with major complaints being issues with the joystick not staying centered as it gets worn in, causing small amounts of drift on the control axes.

5.3.3 Hercules Thrustmaster T16000M

The Hercules Thrustmaster T16000M is a plain looking, but fully featured joystick with 16 buttons, one 8-way POV hat, a throttle slider, and a special hall-effect axis sensor system that provides greater precision than standard analog joysticks. [3]

- *Cost*: Low-mid cost at $65.
- *Control Feature Set*: Exceeds minimum feature set.
- *Ambidextrous Grip*: Yes
- *Reviews*: This joystick gets better than average reviews with most complaints focusing on the joystick feeling cheap while performing admirably.

5.4 Discussion

From a cost perspective, the Saitek Evo wins out, but only barely. The Thrustmaster is a close second, while the X52 is overly expensive at the cost of both of the previous joysticks put together.

All of the joysticks exceed the needed feature sets and are all ambidextrous, so in these categories they're all on an even playing field.

When reviews are factored in, the Thrustmaster is well-regarded whereas the other two get average reviews that aren't too spectacular.

5.5 Conclusion

Once you combine the excellent reviews with the reasonably cheap costs of the Thrustmaster T16000M, it comes out as the clear winner for the joystick category. It is definitely the best performance per dollar. The Saitek Evo is a close second, but the the X52 should not even be considered as an option.

6 References

References

[1] Logitech g saitek x52 flight control system. [Online]. Available: https://www.amazon.com/Logitech-Saitek-Flight-Control-System/dp/B01LY285ZH/ref=pd_day0_147_4?_encoding=UTF8&psc=1\refRID=050PFE0S4QS28T91V2J7

[2] Saitek cyborg evo joystick. [Online]. Available: https://www.amazon.com/Saitek-102989-Cyborg-Evo-Joystick/dp/B0000AW9P1

[3] Thrustmaster t-16000m flight stick. [Online]. Available: https://www.amazon.com/Hercules-2960706-Thrustmaster-T-16000M-Flight/dp/B001S0RTU0