Miniature Mars Rover

DESIGN DOCUMENT

Team Number: sdmay19-09
Client: Dr. Zambreno
Advisers: Craig Rupp

Team Members/Roles:

Calvin McBride: *Hardware/Software Developer,*
Communication,
Status Reporter,
Meeting Note Taker

Sam Westerlund: *Software Developer,*
Communication

Mitchell Freshour: *Hardware Lead*

Douglas Kihlken: *Hardware Developer*

Team Email: sdmay19-09@iastate.edu
Team Website: sdmay19-09.sd.ece.iastate.edu

Revised: 2018-10-13 Version 1

Table of Contents

1 Introduction

1.1 Acknowledgement

This project would not be possible without the support of the Iowa State University ECpE Department, our advisor Craig Rupp, and our client Dr. Joseph Zambreno. The funding, advice, and motivation they provide us has made it possible for us to build and extend the JPL Open-Source Rover, a demanding yet rewarding project. It is with their feedback and support that we hope future teams may benefit from further work on the miniature Mars Rover.

1.2 Problem and Project Statement

In today's evermore complex and demanding job market, potential students and interest groups must make hard decisions on what to study and where to get an education that will help them fulfill their dreams. For students looking to delve into engineering and related disciples, it can be especially difficult to see all of the potential benefits and freedom that an engineering degree can provide them.

To help spur interest in Iowa State University's engineering programs and catch the interest of prospecting students, ISU's ECpE department has devised several display pieces that demonstrate the engineering prowess their senior students wield. One of these projects is the Mars Rover project, a miniature 6-wheeled mars rover model that requires knowledge of software, mechanical, and electrical engineering to construct.

Our team's primary goal is to get the rover (based on JPL's Open-Source Rover project) up and running, with secondary but preferred goals being the use of computer-visual and machine-learning to navigate and possibly handle objects. It is our hope that these features, along with the ability to remote-control the rover, will help catch the interest of visiting students, as well as give future senior design teams a well-built platform to extend for their own projects.

1.3 Operational Environment

The expected operation environment that the miniature rover will operate in is intended to be inside the buildings of the Iowa State University campus. These buildings have hard, smooth floors, clean (non-dusty) air, and a reasonable amount of room to maneuver. While we expect the rover to mostly operate indoors, the JPL Open-Source Rover is capable of operating outside in nice weather. Because of this, the rover may be able to operate outside as long as the weather is nice and users keep the rover on relatively flat terrain.

1.4 Intended Users and Uses

Our end-product is expected to have two different groups of users: student/faculty drivers and senior design teams. The first group consists of potential students and interest groups that will utilize the rover to see first-hand what can be built by Iowa State University engineering students, as well as faculty who will demonstrate the rover's features and move the rover around. The second group are senior engineering students taking Senior Design who are tasked with extending the rover for their own senior design project. This second group is expected to have technical knowledge, whereas the first group is expected to have minimal technical experience.

With these groups in mind, the rover will implement a range of both manual and autonomous navigation in order to "wow" members of the first group, while keeping extendibility and flexibility in mind so that members of the second group may easily build upon the rover.

1.5 Assumptions and Limitations

Assumptions:

- The rover will primarily be operated indoors, in an environmentally-clean area.
- The rover will be remote-controllable, with users able to view what the rover sees through its camera.
- The rover will utilize computer-vision and machine-learning to navigate when not manually controlled.
- The rover will have six independently-powered wheels.
- The rover will use the JPL Open-Source Rover as a base.

Limitations:

- The JPL Open-Source Rover base-build has a battery-life of approximately 4 hours, without additional modifications.
- The rover uses a Raspberry Pi 3, which has a limited number of GPIO pins that are necessary for extending further hardware add-ons.
- The project has a budget of $2,500.
- The JPL Open-Source Rover documentation estimates a minimum of 200 man-hours by experienced teams to build the base rover.

1.6 Expected End Product and Deliverables

The final end-product will be a working mars rover based on the JPL Open-Source Rover project. This rover will utilize computer-vision and machine-learning to move around its environment, as well as remote-control capabilities so that users may operate it manually. The rover will be well-documented and have many GPIO pins so that future teams can work with and extend the rover design. The rover, its documentation, and its source code will be delivered to the client at the end of the Spring 2019 semester, around April/May.

2. Specifications and Analysis

2.1 Proposed Design

We decided to use the JPL Open-Source Rover as the starting point for our miniature rover. Other robot chassis were considered, but JPL's rover was well-documented, open-source, and thoroughly tested, making it ideal for our team. Our rover will most-likely use Intel's Movidius Compute Stick as a small but functional GPU, upon which many computer-vision and machine-learning tasks will be executed. Our team is currently looking into camera systems that will satisfy our computer-vision needs, as well as other sensors such as LIDAR being considered.

The rover hardware/parts as specified in the JPL Open-Source Rove parts-list have been ordered, and its construction will begin once they are received.

Functional Requirements:

- Raspberry Pi should be able to take input from a controller, read and process hardware data, and stream camera visuals to user.
- Working camera for computer-vision to self-navigate.
- Algorithms to detect common indoor objects and people, as well as navigation algorithms to move around such items.
- Working sensors (such as LIDAR, Sonar, Infrared, Camera) to "see".

Non-Functional Requirements:

- Battery life must have a minimum of 2 hours.
- Rover should take no more than 1 second to respond to manual user input.
- Rover should take an average of 5 seconds to scan, process, and create a plan for navigating its environment, with a worst-case time of 10 seconds.
- Remote-control signals should reach several meters, ideally throughout most of the a given building.

2.2 Design Analysis

As of the writing of this document (Version 1, October 13, 2018), no technical analysis has been possible as we do not have hardware to test on. However, the JPL Open-Source Rover project has been tested and verified by a much more technically knowledgeable and experience team of engineers than us, and we feel an in-depth analysis of the base rover design is not necessary (beyond some testing).

For the components we will personally be developing and extending the rover with (computer-vision, machine-learning, etc.), in-depth testing and analysis will have to wait until hardware is received and much of the base rover is constructed. However, we have compiled what we believe are accurate strengths and weaknesses in our proposed design:

Strengths:

- Well-tested and thoroughly vetted rover base/starting point.
- Relatively cheap and easy to understand.
- Several sensors for accurate and thorough environmental analysis by rover.
- Raspberry Pi is well-documented, making tasks like computer-vision and machine-learning easy to start and develop with.

Weaknesses:

- Many electrical components are not vacuum-sealed, making them susceptible to bad weather and water.
- Computer-vision is a relatively young field in computer science, making it a somewhat difficult task to implement for our junior team.
- Computer-vision and machine-learning is a relatively high cost in terms of processing power, making the rover's 3-4-hour battery life even shorter.

3 Testing and Implementation

3.1 Interface Specifications

The Raspberry Pi will communicate with several hardware entities, such as the camera, LIDAR sensor, wheel motors, and onboard GPU stick. In addition to this, the Raspberry Pi will have to connect to a remote controller (whether that will be a simple console controller or a website/web-interface is yet to be decided) and stream video to the user wirelessly.

Testing these interfaces will involve observing that the hardware performs actions as they are instructed (cameras turn when instructed, wheels rotate at specified speeds, etc.) and manually confirming that data being received from sensors is correct (ensuring the camera is working and streaming video, manually outputting datapoints from sensors, verifying the rover moves when remote-controller is used, etc.). Other means of verifying the accuracy of sensors and commands will be devised once hardware is received and the base rover construction has begun.

3.2 Hardware and software

Testing our design will utilize several hardware and software testing suites. Hardware testing will most likely be done with equipment located in Coover Hall, and software testing will be done on university workstations or personal computers using software provided by the university or that is free/open-source. Hardware tools will include multimeters and oscilloscopes, while software tools will include IDEs and debugging software.

3.3 Process

The first phase of testing we will do will be simple mechanical testing on the rover body. This will be done by setting various weights on the body to ensure it does not break, moving body parts to ensure they do not break off and have the expected range of motion, and pushing the body around to ensure the wheels and joints are operating optimally.

The second phase of testing will involve the electrical component after integrating it into the body. This will mostly involve insuring the various electrical components are getting power and turn on. Other testing (such as ensuring correct data transmission power levels) will wait until after software has been integrated.

The third stage of testing will involve static testing and analysis of software using an IDE on a computer, independent of physical hardware. This is the most basic form of software testing and is functionally agnostic (i.e. ensures that all software components "fit" together properly, not necessarily that they function correctly).

The fourth stage of testing will involve integrating the software into the rover. In contrast to statically analyzing and testing the software as in stage three, this step involves ensuring the behavior of the software correctly models how we want the hardware to run. Tests in this stage will involve reading hardware data, sending hardware commands from the Raspberry Pi, data processing, and other tests to ensure the hardware and software are properly interfacing.

Final testing will involve miscellaneous tests that weed out extreme or unlikely behavior, as well as ensuring that extending the rover can be done in a relatively simple manner.

[A flow diagram will be added to this document in Version 2.]

3.4 Results

[This section is not currently applicable as we do not have any testing data yet. This section will be updated in a later version to reflect the results of our testing.]

4 Closing Material

4.1 Conclusion

Our project goal is to have a working miniature rover based on the JPL Open-Source Rover that demonstrates the skills and technical knowledge wielded by Iowa State University senior engineering students in order to impress and draw in potential students for the ECpE department. With the JPL Open-Source Rover as a base, we plan to extend it with computer-vision and machine-learning to enable both manual and autonomous navigation, as well as ensure that future senior design teams can work on and extend our rover. Our solution demonstrates the utilization of two fields of computer science that are becoming increasingly popular, as well as gives future engineering teams plenty of room and flexibility when building upon our design.

4.2 References

[1] Github (2018, Sept.). NASA JPL Open-Source Rover [Online]. Available: https://github.com/nasa-jpl/open-source-rover [Accessed Oct. 13, 2018]