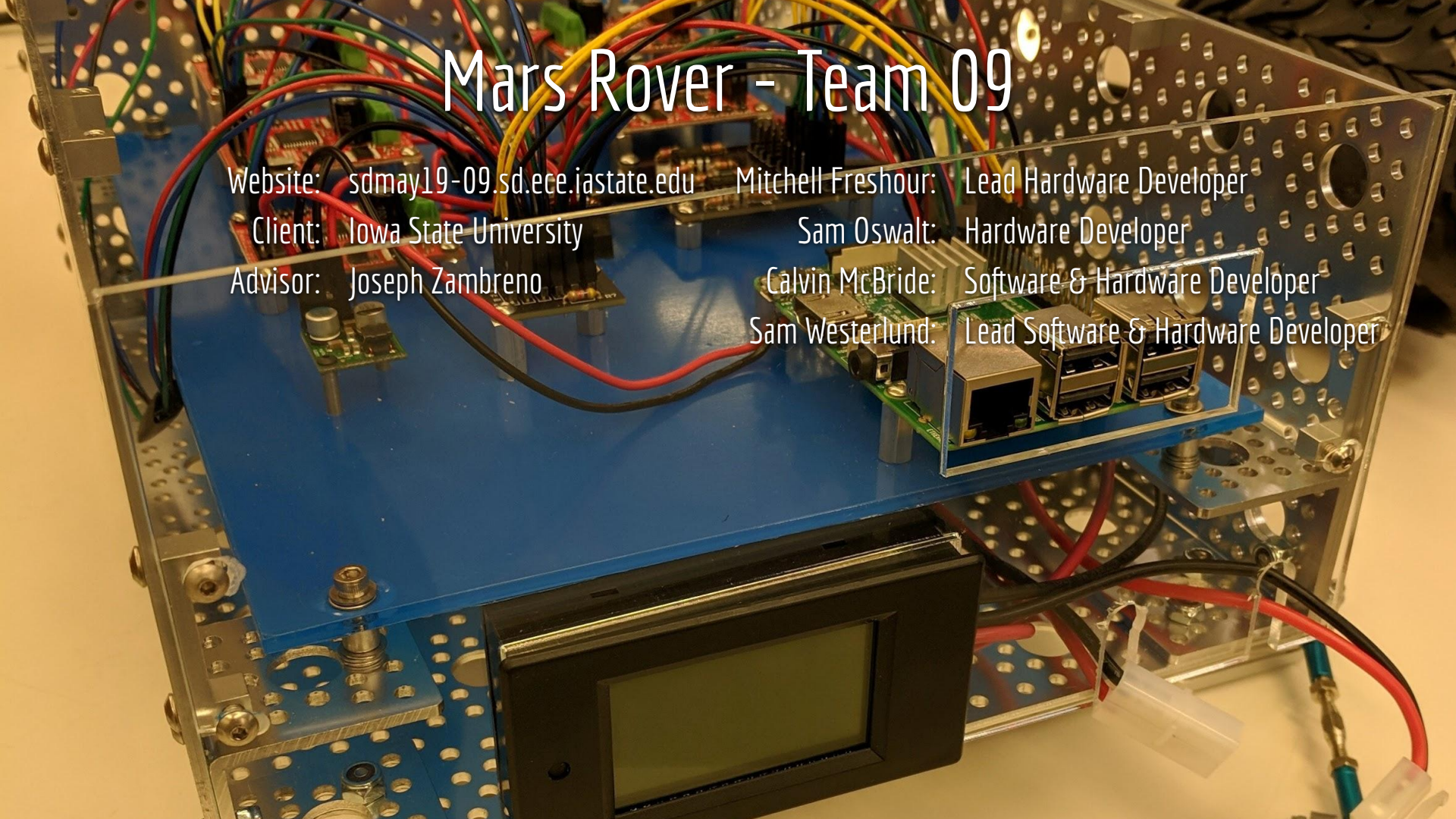
Mars Rover - Team 09



Website: sdmay19-09.sd.ece.iastate.edu

Client: Iowa State University

Advisor: Joseph Zambreno

Mitchell Freshour: Lead Hardware Developer

Sam Oswalt: Hardware Developer

Calvin McBride: Software & Hardware Developer

Sam Westerlund: Lead Software & Hardware Developer

Problem Statement

Problem: Iowa State University's ECpE department needs a show piece to capture the attention of and engage potential students and visitors at Iowa State University.

Solution: Construct a well-documented and extensible miniature Mars rover that can either be remote-controlled or navigate itself using machine-learning and computer-vision.

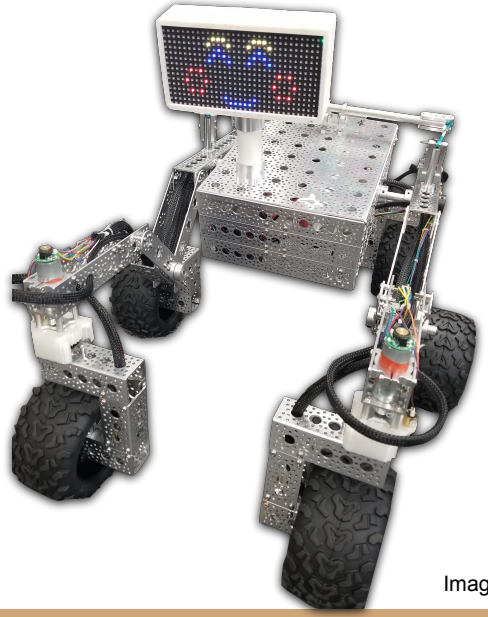


Image Source: NASA JPL. https://github.com/nasa-jpl/open-source-rover

Project Design

- Key Features
  - Computer Vision
  - Easy to start, stop, and manage
  - Easy to extend/add to for future teams
- Components
  - JPL Open-Source Rover software (github.com/nasa-jpl/open-source-rover)
  - Camera and LIDAR sensor

Image Source: NASA JPL. https://github.com/nasa-jpl/open-source-rover

Use Cases

- The rover is shown off by Iowa State staff or student to those touring the university
- Senior engineering students taking Senior Design extend the rover’s functionality for their own senior design project

Functional Requirements

- Raspberry Pi reads input from an external web interface, controls hardware, and processes data.
- Camera streaming and labeling of objects in view to the user.
- Working camera for computer-vision.
- Working sensors LIDAR and camera to detect and map it’s surroundings.
- Implementation of a neural network to detect common indoor objects and people.
- The end will manually control the Mars Rover through the browser at the same page of the camera input. This page will have simple buttons that will move the rover in that direction.

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.
- Remote-control signals should reach several meters, ideally throughout most of the a given building.
- Python will be used as the programming language to control the hardware. React will be used as the web interface.
- The rover will be able to handle inclines at 15 degrees, and uneven terrain where the disparity from the left to right wheels is less than 1 inches.

Technical Constraints & Considerations

- 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, and our team has limited electrical and mechanical experience.

Detailed Design

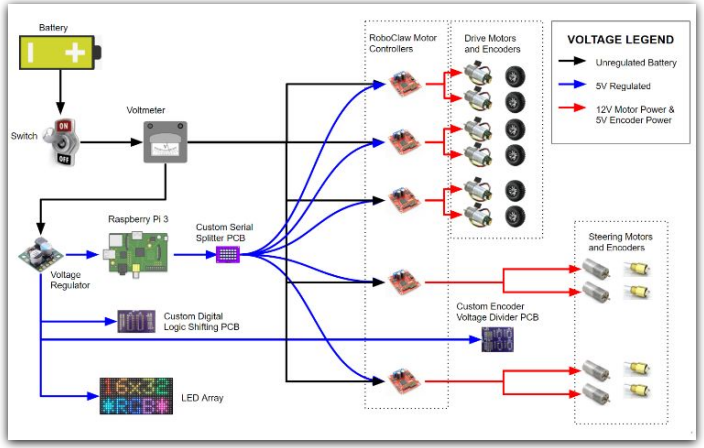


Image Source: NASA JPL. https://github.com/nasa-jpl/open-source-rover

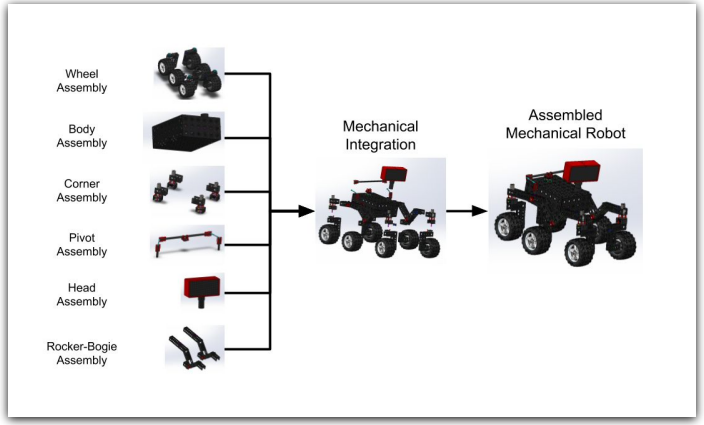


Image Source: NASA JPL. https://github.com/nasa-jpl/open-source-rover

Client

React

Tornado

SLAM

MobileNet V2

Keras

TensorFlow

python™

RaspberryPi

Implementation (Software)

All the software runs on the Raspberry Pi. There are two major components to the software, the UI service and the Rover control service.

Client:

- Any browser that supports HTML5 and has JavaScript enabled

UI Service:

- React.js

Rover Control Service:

- Python
  - Keras
    - MobileNet V2
  - Tensorflow
  - Tornado (websockets)
  - SLAM
- SSH

Implementation (Power Delivery)

- Power:
  - 77Wh lithium ion battery
  - Voltmeter to display information to operator
  - Terminal block to split power to motor controllers
  - Voltage regulator to send 5V to Raspberry Pi for power
- Motors:
  - 10 Brushed 12V DC motors
    - 6 drive motors, 4 corner motors
  - 5 RoboClaw Motor Controllers
    - 3 drive boards, 2 cornering boards

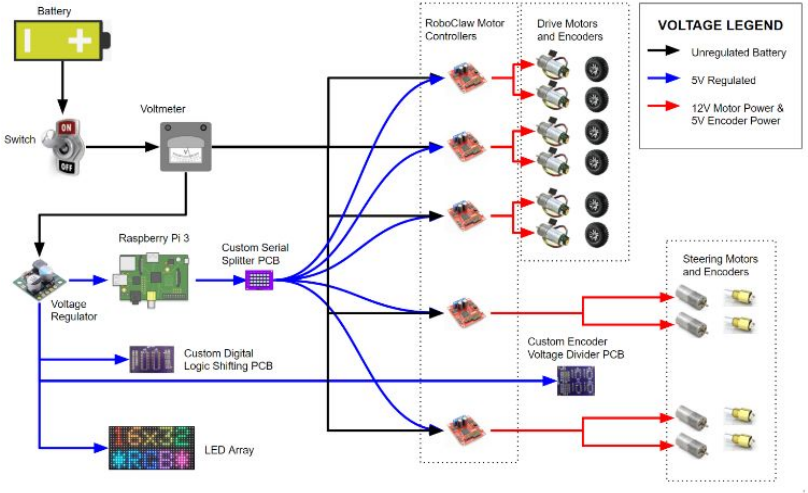


Image Source: NASA JPL. https://github.com/nasa-jpl/open-source-rover

Implementation (Data Transfer)

- Data Transmission:
  - Serial Splitter PCB
    - Splits commands sent by raspberry pi amongst motor controllers
  - Motor Controllers
    - Reads 5V TX/RX signal for drive motors
    - Reads 2V encoder position from voltage divider PCB
  - Voltage Divider PCB
    - Translates 5V signal from absolute encoder to 2V signal to motor controller
  - LIDAR
    - Transmits point data to Raspberry Pi
  - PiCamera
    - Sends camera images directly to Raspberry Pi

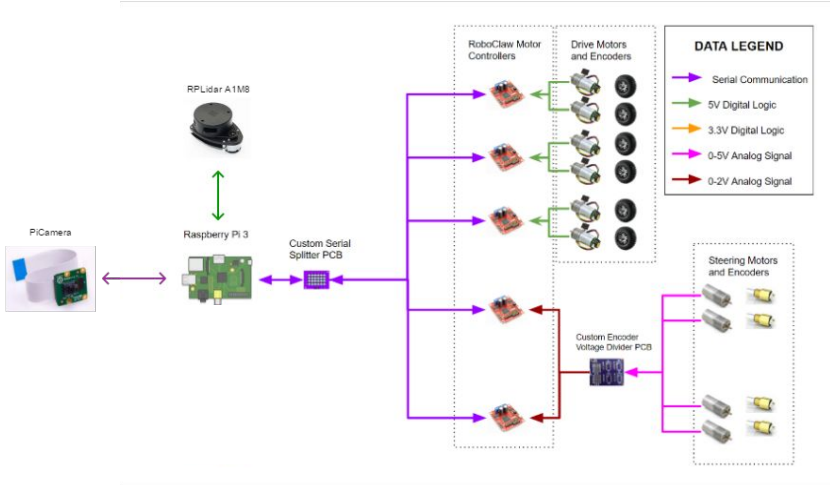


Image Source: NASA JPL. https://github.com/nasa-jpl/open-source-rover

HW/SW/Technology Platforms Used

- React.js
- Python
- Tornado (python sockets)
- Google Maps
- Chart.js
- Npm
- Keras (TensorFlow backend)
- Raspi
- Intel Movidius Compute Stick
- RPLidar
- RoboClaw Motor Controllers
- OshPark PCBs

Image Source: RPLIDAR

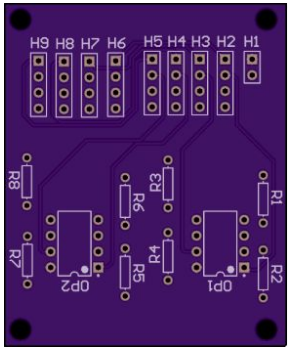


Image Source: NASA JPL.
https://github.com/nasa-jpl/open-source-rover

Unit & Interface Testing

- System testing and code reviews on newly created software
  - User was able to connect to the web interface and view the camera strea, classifications, graphs, and all other implemented features

System Integration Testing

- Power System
  - Multimeter verification during integration
- Motor Controller Calibration
  - Conducted in accordance to JPL Motor Calibration documentation[1]
- PCB Testing
  - Conducted in accordance to JPL PCB Testing documentation[2]
- Functionality Testing
  - Conducted upon full integration by verifying individual rover functions

Roboclaw Calibration[1]: https://github.com/nasa-jpl/open-source-rover/blob/master/Electrical/Calibration.pdf

PCB Testing[2]: https://github.com/nasa-jpl/open-source-rover/blob/master/Electrical/PCB%20Testing.pdf

Task Responsibility

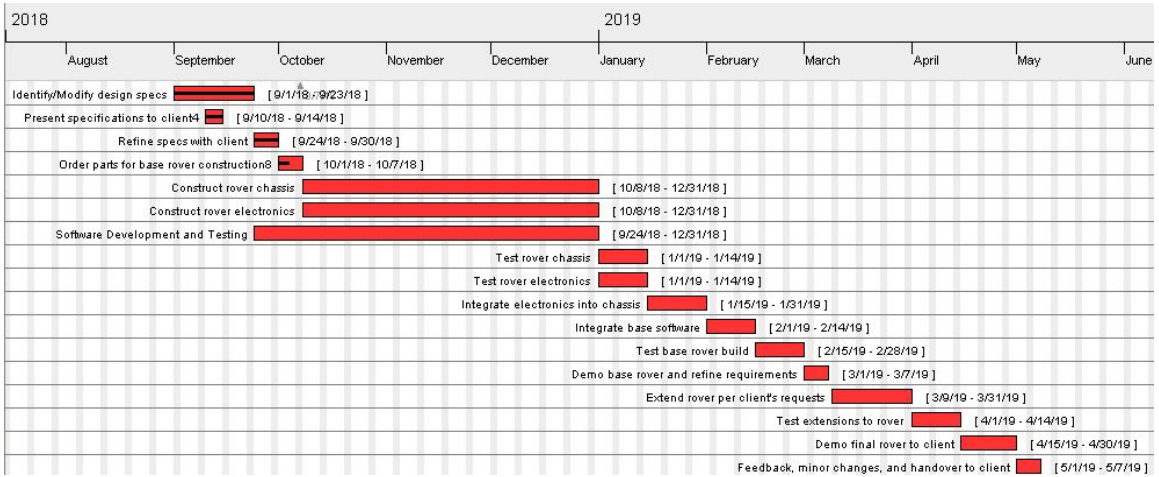
Two members focused on wiring, testing, and calibrating components, another member focused on machining and assembling components, and the final focused on software development and assembly.

- Hardware
  - Mitchell - Electrical build wiring and soldering. Calibration of motor components and electrical testing.
  - Calvin - Body machining and fabrication, laser cutting, and main body assembly. Assisted Mitchell with calibration and testing after body was assembled.
  - Sam Oswalt - Electrical assembly and electrical testing.
- Software
  - Sam - Camera and LIDAR integration, prototype UI with controls, sensor charting, basic image recognition, google maps integration. Rover assembly and some wiring.

Planned Schedule

Our initial deadline for the rover's final assembly was March of 2019. Issues with calibration, needing to re-order parts, and unexpected complexity of the project pushed this deadline back resulting in the rover assembly only being completed at the end of April, leaving minimal time for testing or further software development.

Identify/Modify design specs	9/1/18	9/23/18
Present specifications to client4	9/10/18	9/14/18
Refine specs with client	9/24/18	9/30/18
Order parts for base rover construction8	10/1/18	10/7/18
Construct rover chassis	10/8/18	12/31/18
Construct rover electronics	10/8/18	12/31/18
Software Development and Testing	9/24/18	12/31/18
Test rover chassis	1/1/19	1/14/19
Test rover electronics	1/1/19	1/14/19
Integrate electronics into chassis	1/15/19	1/31/19
Integrate base software	2/1/19	2/14/19
Test base rover build	2/15/19	2/28/19
Demo base rover and refine requirements	3/1/19	3/7/19
Extend rover per client's requests	3/9/19	3/31/19
Test extensions to rover	4/1/19	4/14/19
Demo final rover to client	4/15/19	4/30/19
Feedback, minor changes, and handover to client	5/1/19	5/7/19



Risks & Mitigation

- Many electrical boards 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.
- Errors in JPL documentation
- Team's lack of knowledge in electrical and mechanical engineering

Lessons Learned

- Importance of coordinating schedules.
- It can be hard to give accurate schedules and estimates.
- Technical planning related to electrical engineering and mechanical assembly.
- We were more productive and accountable when we all met together regularly.

Future Work

For future work on software, future teams should be able to build on top of the current design.

Necessary for full function:

- Replace and calibrate right side corner motors

Software improvements may include:

- Implementing GPS navigation
- Autonomous driving
- UI improvements

Closing Remarks

- Our team set out to build a miniature mars rover that utilized computer vision to help autonomously navigate its environment.
- Long-term design goal was to be well documented and easily extensible for future senior-design teams.
- We feel that we did not adequately achieve the goal of autonomous navigation utilizing the camera or LIDAR system, but we do feel we have left the project in such a way that it can be easily completed and extended by future teams.
- We still learned a lot about electrical engineering and assembly, team management, project timeline estimation, and managing a long-term project from start to finish.