**Logo%20Main%20200**

**Boston University**

**Electrical & Computer Engineering**

**EC463 Senior Design Project**

**First Prototype Test Report**

**Mars Rover: Autonomous Navigation**



by

Team #28

Rover Boi

Team Members

Daniel Crawley crawley@bu.edu

Brian He brianhe@bu.edu

Tommy Lam tlam11@bu.edu

SeungYeun Lee sylee538@bu.edu

Linden Vo [lindenvo@bu.edu](mailto:lindenvo@bu.edu)

November 15, 2019

To: Alan Pisano, Michael Hirsch, Osama Alshaykh

**Required Materials**

Hardware:

* 1/8th 4WD Electric Power R/C Rock Crawler
* ESP32 Micro Controller
* Battery(7.2V,2000mAh) 51C00-03200
* Intel RealSense D415 Depth Camera
* TFMini - Micro LiDAR Module
* LIDAR lite v3
* Servo(9KG)(Waterproof) 51C00-SP9002
* Electronic Speed Controller (Waterproof) 98120
* Motor (RC540) W/ Gear (12T) 03012

Software:

* Windows 10 Operating Systems
* Visual Studio Community 2019
* Espressif
* ESP32 Script
  + Calibrate the motors of the rover
  + Drive motors and avoid obstacles
* Intel RealSense Librealsense Library
  + Realsense Viewer

**Setup**

Our equipment set up is the crawler itself with all necessary sensors and controllers attached onto the top of the Rover. We have the MaxSonar Ultrasonic Range Finder mounted on the front of the vehicle. It will detect when our Rover reaches a wall and will stop accordingly. We also have two TFMini Micro Lidars mounted on the left side of our Rover. It will serve as the basis of the steering logic for our Rover, for now. Depending on the readings from the two MicroLidars, our servos will either turn left, right, or go straight and will avoid collision. We also have a separate Intel RealSense Camera D415 Depth Camera with many built in functionalities.

**Pre-testing Setup Procedure:**

ESP32 (if Espressif is not installed on windows)

* Follow the instruction as shown in this website (<https://docs.espressif.com/projects/esp-idf/en/latest/get-started/index.html>)

ESP32 (Espressif installed on windows)

1. Make sure all ESP32 GPIO are connected correctly.

Please make sure Rover is wired correctly as per the diagram:



Key:

1. Servo 51C00-SP9002
2. LIDAR lite v3 module
3. TFMini - Micro LiDAR Module - “Sensor 1”
4. ESP32
5. 7-segment display
6. TFMini - Micro LiDAR Module - “Sensor 2”
7. Optical Encoder
8. Front Stepper Motor
9. ESC
10. Rear Stepper Motor

ESP32:

*cd into* folder ec463 and run *idf.py -p [PORT] flash monitor*

Intel RealSense Camera:

From terminal with librealsense installed in installation directory, run ./realsenseviewer

**Hardware Pinout**

|  |  |
| --- | --- |
| **ESP32 Pin #** | **Usage/Description** |
| A5 (GPIO #4) | Optical encoder (PCNT pulse input) |
| 14 (GPIO #14) | UART\_TX for left side front micro lidar |
| SCK (GPIO #5) | UART\_RX for left side front micro lidar |
| 27 (GPIO# 27) | UART\_TX for left side back micro lidar |
| 15 (GPIO#15) | UART\_RX for left side back micro lidar |
| SCL (GPIO #22) | i2c clk for LiteLidar V3 |
| SDA (GPIO #23) | I2C data for LiteLidar V3 |
| SCL (GPIO #22) | i2c clk for alphanumeric display |
| SDA (GPIO #23) | I2C data for alphanumeric display |
| MOSI (GPIO #18) | esc (PWM0A) |

**Testing Procedure**

1. Place the Rover onto the ground at least 1 feet away from and parallel to the wall.
2. Check to make sure the sensors are working as expected by covering individual sensors.
   1. Covering Sensor 1 should turn the front wheels right.
   2. Covering Sensor 2 should turn the front wheels left.
   3. Covering front sensor should stop the vehicle.
   4. Can use serial port and terminal to debug if necessary to determine if measurements are accurate
3. Turn on the ESC on the right side of the Rover.
4. Power ESP32 with external 5V power bank.
5. After three seconds, Rover will begin moving.
6. Monitor Rover and make sure it meets all measurable criteria.

**Overview of Steering Logic**



**Measurable Criteria**

Criteria for successful running and output are like such:

1. The Rover should successfully move parallel to the wall.
2. Rover should be able to move at 0.1 m/s - 0.5 m/s.
3. Rover should be able to accurately detect the distance of objects in front of it and stop within 20cm of the object.
4. Rover should be able to steer and avoid collision when detected. It should change direction when an object is detected on the side or when it is in a corner.

**Testing Results/Observations**

|  |  |
| --- | --- |
| **Objectives** | **Complete? (Y/N)** |
| Travel 5m | Y |
| Travel parallel to the wall | Y |
| Travel at a constant speed | Y |
| Stop before hitting a wall | Y |
| Steer to avoid collision | Y |
| RealSense Camera able to detect and return distance | Y |
| RealSense Camera able to detect People | Y |

**Test Report**

The prototype testing allowed us to validate the goals defined in our test plan, in which all of them were met. The rover was able to travel five meters on its own down a hallway while parallel to a wall using MicroLIDAR. The average speed displayed on the seven segment alphanumeric display on our crawler showed less than 0.5 meters per second and at a constant ~0.1 meters per second. For future improvements we can increase the speed control of the vehicle to be more responsive even while travelling at 0.5 meters per second. The rover was able to detect a wall with the LiteLIDAR module on the front of the vehicle and stop within 20 centimeters. The rover also would correct itself to remain in the center of the path parallel to the wall at a specific distance from the wall. When working with the RealSense Camera, the camera is able to detect and return the distance to an object, as well as utilize facial recognition and detection of people as the first step in obstacle detection and avoidance.

**Conclusions**

During our test, we received some valuable feedback from the course staff. They recommended that we use ultrasonic sensors on the sides to detect walls, instead of the MicroLIDAR sensors we used. We initially stayed away from that approach due to the potential for interference between ultrasonic sensors, but the staff pointed out that we could stagger their use so only one is on at a time. Another piece of advice given had to do with the RealSense camera. Teams in previous years found that their unit does not work well outdoors. After some research, we found that Intel has published some guides for tuning the camera for outdoor use. Lastly, course staff also informed us of how difficult it is to use a Jetson TX2 as one would an Arduino or Raspberry Pi. We have been experiencing some of those issues, and so have used an ESP32 for interfacing with the ESC and servo as well as some sensors, and plan on having the TX2 be responsible for the more computationally intensive tasks.

In conclusion, we were able to meet the goals defined in our test plan. However, in comparison to our original goals of traversing through an aperture and to certain locations given GPS coordinates as well as acquiring and following an ARTag, the team still has a lot more goals to accomplish. In the future, the plan is to use the RealSense D415 Depth Camera with the Jetson TX2 to detect an ARTag and localize its coordinates before navigating towards it. The TX2 will need to send messages to the ESP32 in order to control the crawler. Despite the design team’s shortcomings, morale is high with current accomplishments and work will continue throughout the semester to improve and add functionality to the design that satisfies the overall requirements of the project.