**CROOMBA Datasheet**

**Operation**

The CROOMBA requires between a 12V and 20V battery with an amp-hour recommended minimum of 3 Ah for about 30 minutes of operation. The lever connectors for the positive and negative terminals of the battery may be found along lines 2 and 5. Refer to *Figures 1* through *2* for locating the line numbers.

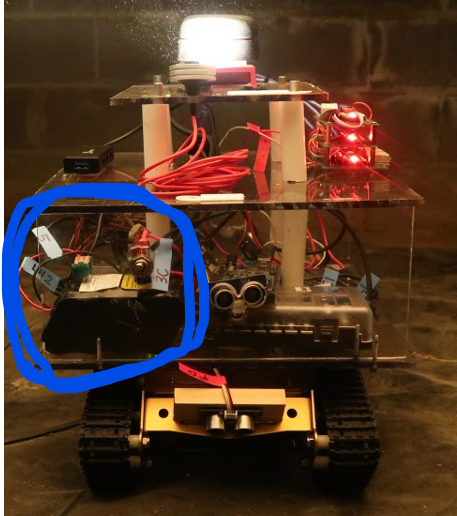


Figure Location of Battery

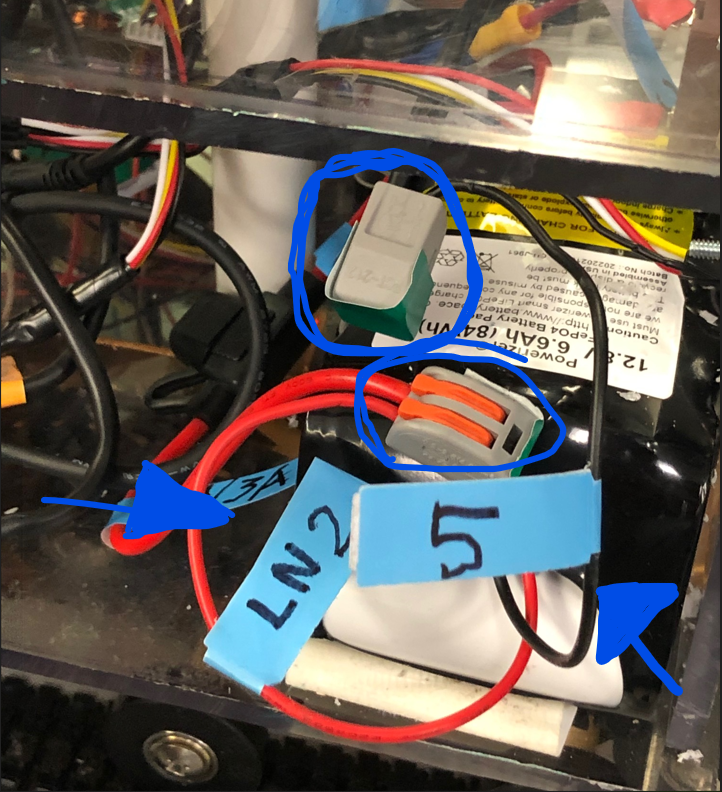


Figure Line Number Locations

Remote access to the CROOMBA is possible with the following steps:

1. Connect the to the CROOMBA’s WiFi network named: CapstoneRobotTeam1
2. Using Windows Desktop Remote:
   1. Enter IP address of Pi: 192.168.0.100
   2. Log in
      1. User: *ubuntu*
      2. Pass: *Croomba*
3. Once inside run the necessary scripts documented in the Robot.Setup GitHUB file
4. Run Combined\_Code.py
5. Launch Arduino IDE
   1. Open Serial Monitor
      1. Enter Manual Commands

In addition to remote access the CROOMBA’s program may be accessed locally via a monitor, keyboard, and mouse attached to the ports shown in *Figure 3* through *4*

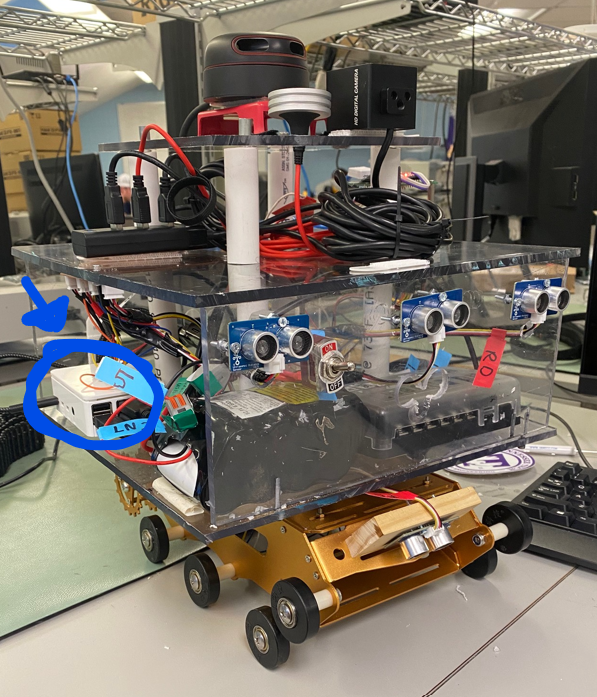


Figure Location of Raspberry Pi

It is recommended that the movement tracks be removed as shown in *Figure 3* to handicap the Croomba for safety.

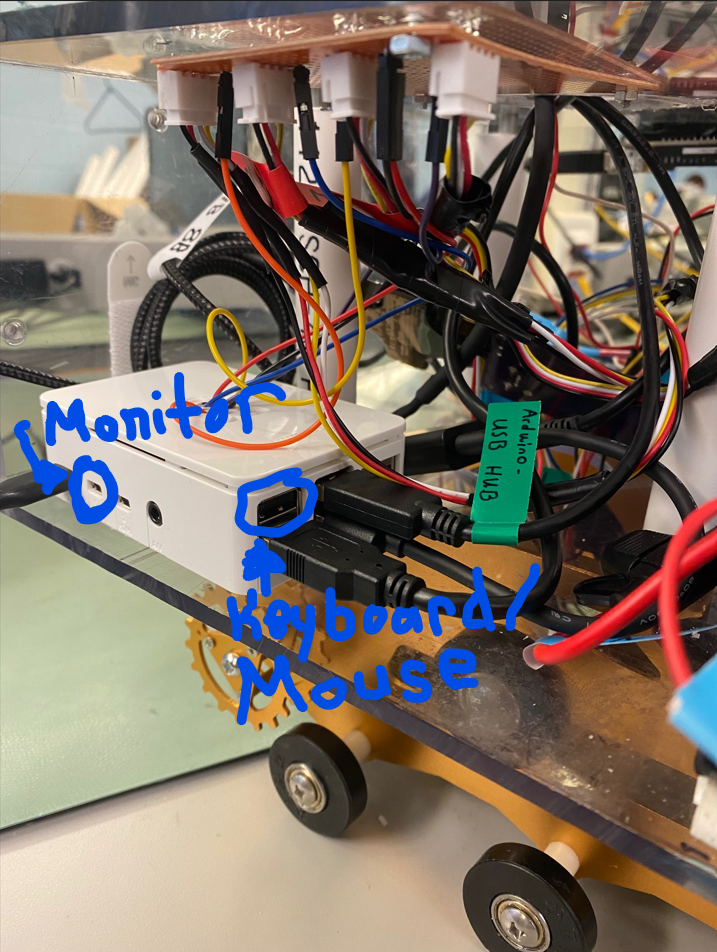


Figure Open Ports for Local Access

**Experimentation and Results**

*Table 1 Measures of Success*

|  |  |  |
| --- | --- | --- |
| Measure of Success | Target Value | Achieved Value |
| Battery Life | 1 Hour | 1.1 Hours |
| Map Percentage of Crawl Space | 70% | 84% |
| Incline Mobility Test | 30° | 15° |
| Environmental Data | Recorded | Recorded |

**Power**

In order to verify the power subsystem is meeting all requirements, 5 trial runs measuring the voltage and current of each main subline coming off the battery were carried out, and the results are in the tables below. We ran over an hour and fifteen minutes of trial runs while programming. The power system successfully powered all subsystems without outside assistance. This proves the two constraints laid out in the proposal that the power subsystem will provide correct current and voltage to the subsystems for a time that would allow mapping of a crawlspace environment.

**Main Power LN 2 & 5**

| **Trial Number (N)** | **Current(A)** | **Voltage (V)** |
| --- | --- | --- |
| 1 | 0.526 A | 13.26 V |
| 2 | 0.627 A | 13.26 V |
| 3 | 0.675 A | 12.85 V |
| 4 | 1.265 A | 12.68 V |
| 5 | 1.326 A | 12.75 V |

**Navigation LN 7A & 8A**

| **Trial Number (N)** | **Current(A)** | **Voltage (V)** |
| --- | --- | --- |
| 1 | 0.7 A | 5.33 V |
| 2 | 0.6 A | 5.34 V |
| 3 | 0.79 A | 5.33 V |
| 4 | 0.7 A | 12.68 V |
| 5 | 1.326 A | 12.75 V |

**Network Router LN 9A & 10A**

| **Trial Number (N)** | **Current(A)** | **Voltage (V)** |
| --- | --- | --- |
| 1 | 0.1 A | 5.3 V |
| 2 | 0.12 A | 5.32 V |
| 3 | 0.12 A | 5.31 V |
| 4 | 0.28 A | 5.31 V |
| 5 | 0.33 A | 5.3 V |

**Movement LN 11A & 12A**

| **Trial Number (N)** | **Current(A)** | **Voltage (V)** |
| --- | --- | --- |
| 1 | 0.526 A | 13.26 V |
| 2 | 0.627 A | 13.26 V |
| 3 | 0.675 A | 12.85 V |
| 4 | 1.265 A | 12.68 V |
| 5 | 1.326 A | 12.75 V |

**Environmental Sensing LN 13A & 14A**

| **Trial Number (N)** | **Current(A)** | **Voltage (V)** |
| --- | --- | --- |
| 1 | 0.0642 A | 5.24 V |
| 2 | 0.0645 A | 5.23 V |
| 3 | 0.0644 AA | 5.26 V |
| 4 | 0.0.065 A | 5.31 V |
| 5 | 0.064 A | 5.3 V |

**Navigation**

**The Model Crawlspace**

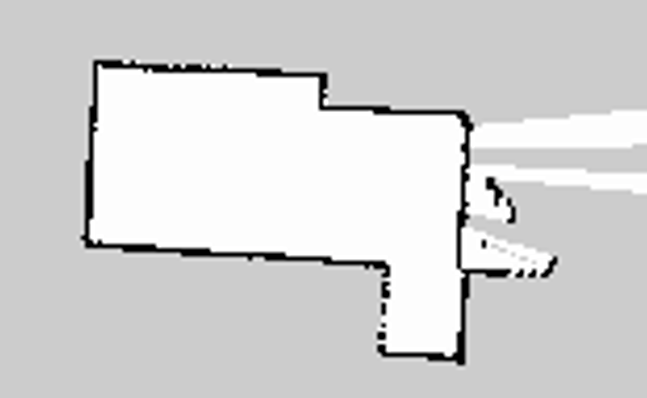
[](https://github.com/ECE-4961-Capstone-Team-1-Spring-Fall-22/Autonomous-Crawlspace-Inspection-Robot/blob/main/ExperimentalPictures/ModelCrawlspace/Model_Crawlspace.jpg) To test the navigation functionality of the autonomous crawlspace inspection robot, a rectangular-shaped model crawlspace was created, with a length of 16.5 ft, a width of 6 ft, and a resulting area of 99 square feet. Floor tiles were used as a reference for quantifying the crawlspace area, with each floor tile having an area of 9 in by 9 in. The walls of the model crawlspace were made with various materials available in the laboratory and had a minimum height of 17 in and a maximum height of 36 in.

**Mapping**

The main job of the system's lidar is to create a map of the surrounding area without having to be provided a manually drawn map by the operator. To ensure that the map is being generated correctly, the perimeter of our test area was found by measuring the picture at a set zoom and creating a proportion from that value, and the percent error was calculated. The areas missed were then measured on the screen and the proportional value was applied to convert into the physical distance. The focus of this experiment was to identify the areas along the perimeter that the lidar missed as the additional points it caught past the perimeter are blocked by the true wall.

| **Perimeter (ft)** | **Measured Perimeter (ft)** | **Percent Error (%)** |
| --- | --- | --- |
| 47.75 | 45.74 | 4.39 |
| 41 | 38.63 | 6.14 |
| 41 | 38.68 | 6.00 |

The results show that the lidar was able to successfully map the majority of the perimeter as it navigated throughout the crawlspace. In the worst case, the robot only failed to map 6.14% of the model environment. However, this percentage should be mostly negligible because none of the gaps in the walls of the model crawlspace are large enough for the robot to fit through.

[](https://github.com/ECE-4961-Capstone-Team-1-Spring-Fall-22/Autonomous-Crawlspace-Inspection-Robot/blob/main/ExperimentalPictures/ROSGeneratedMaps/ROS_Crawlspace_Map_V1.PNG)

**Maneuvering**

In adherence to movement specifications, the robot needs to be capable of maneuvering through 70 % of the model crawlspace. To test adherence to this requirement, the team members counted which of the floor tiles in the model crawlspace the robot had not touched, and therefore how much of the total area of the crawlspace the robot had not maneuvered through. Additionally, the robot was initially placed in different corners of the model crawlspace to test the behavior from different initial positions.

In this experiment, the total area of the crawlspace was roughly 100 square feet and the total area covered by the robot is calculated and listed, along with the total percentage of the crawlspace area covered during the test. It is worth noting that in addition to the area listed in the table below, the robot often crossed back through past squares, effectively increasing the total area covered by the robot.

| **Area Where Present (ft^2)** | **Percent of Area Covered (%)** |
| --- | --- |
| 86.50 | 86.50 |
| 91.00 | 91.00 |
| 82.00 | 82.00 |
| 76.38 | 76.38 |
| 85.94 | 85.94 |

As shown in the above table, the crawlspace inspection robot was able to maneuver through more than 70 % of the model crawlspace area, often maneuvering through more than 80 % of the space without getting stuck or needing assistance from the operator. It is important to note that while the robot did not enter 100 % of the tiles in the crawlspace area, not every tile must be entered for an incredibly detailed map to be created within the ROS software using the LiDAR sensor.

**Imaging Quantity**

After adjusting the project's scope, imaging has been reprioritized and will take precedence over autonomous control. Because we plan to stitch the images of the crawlspace together using python and OpenCV, it is essential to test what percentage of the ceiling we can successfully photograph. We tested this metric by treating the photographs taken as a jigsaw and then finding the area of any spots not recorded and comparing that area to the overall ceiling area.

| **Total Ceiling Area (ft^2)** | **Pictures Taken** | **Area Pictured (ft^2)** |
| --- | --- | --- |
| 87 | 33 | 7920 |
| 87 | 36 | 8640 |
| 87 | 54 | 12960 |
| 87 | 51 | 12240 |
| 87 | 42 | 10080 |

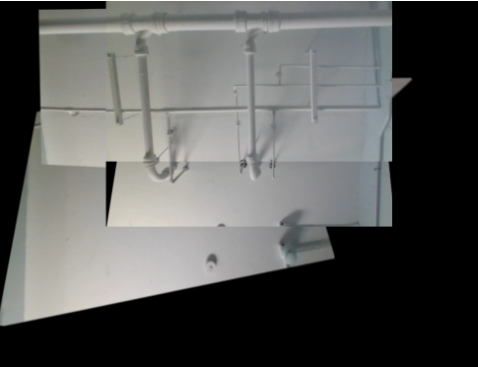
After allowing the robot to run its course, the number of pictures taken can be seen in the table above. The real area captured by each picture was then multiplied by the number of pictures taken to find a value for the full area pictured. While this value will feature some overlap, it is significantly large enough to convince the team that the robot saves pictures that will cover the entire ceiling area. Along with this metric, the photos were flipped through and ceiling landmarks were used to confirm that no part of the ceiling was missed.

**Imaging Quality**

Python code was created using the OpenCV library to perform the image stitching. This test was performed using both real pictures taken of the ceiling using the robot's camera and pictures of a crawlspace ceiling. This allows us to confirm that the code will work both in a real crawlspace setting and will also work when using the camera.

| **Compared Pictures** | **Percent Overlap (%)** |
| --- | --- |
| Test 1, 1 and 2 | 56 |
| Test 2, 1 and 2 | 55 |
| Test 2, 3 and 4 | 45 |
| Test 2, 5 and 6 | 52 |

Tests were performed on the pictures taken to find the necessary overlap percentage between pictures. It was found that this overlap was around 50% for all the stitchings that worked correctly.

[](https://github.com/ECE-4961-Capstone-Team-1-Spring-Fall-22/Autonomous-Crawlspace-Inspection-Robot/blob/main/ExperimentalPictures/Camera2_Results.png)

**Wireless Access Point**

The main job of the system's wireless access point is to create a wireless network for communication between the system's main control, the Raspberry Pi, and the system operator in the case that manual control is needed. Adhering to the given constraints, the wireless access point operates on 2.4 GHz and is a private network that requires a password for use. To ensure that the wireless access point was sufficient for use, the distance and overall connectivity, and control were tested four times.

In each of the four tests, a laptop was connected to the Raspberry Pi at a distance of 100 feet, first on the fourth floor of Brown Hall and then on the third floor of Brown Hall. The robot was pinged by the laptop with 32 bytes of information, and the response time in milliseconds from the computer to the Raspberry Pi and back was recorded. Finally, the overall system control was tested by opening applications and interacting with the system peripherals such as one of the USB cameras over Microsoft Windows Remote Desktop Protocol.

| **Distance From Robot (ft)** | **Average Ping Time (ms)** | **Full Desktop Control?** |
| --- | --- | --- |
| 100 | 4 | Yes |
| 100 | 3 | Yes |
| 100 | 3 | Yes |
| 100 | 3 | Yes |

The results show that the wireless access point is sufficient for the application of the crawlspace inspection robot and that operators will be able to successfully control the robot from well over 70 feet away, allowing for manual control in more significant than average home crawlspaces, which would only require a max communication distance of 70 feet.

**Environmental Sensing**

**Temperature**

The sensor was placed outside at different times and compared to the ambient temperature given by Weather.com. The test was performed like this because the commercially available sensor turned out to be incredibly slow and inaccurate. It was also done in this way because an outdoor environment is where the robot will be used.

| **Date/Time Recorded** | **System Temperature Sensor (°F)** | **Website Listed Temperature (°F)** | **Percent Error (%)** | **Difference (°F)** |
| --- | --- | --- | --- | --- |
| 11/5/2022 23:30 | 62.825 | 63 | 0.28 | 0.175 |
| 11/6/2022 0:44 | 61.5375 | 61 | 0.88 | 0.5375 |
| 11/6/2022 1:23 | 57.925 | 59 | 1.82 | 1.075 |
| 11/6/2022 11:53 | 62.2375 | 64 | 2.75 | 1.7625 |
| 11/6/2022 15:06 | 71.125 | 70 | 1.61 | 1.125 |
| 11/6/2022 16:33 | 66.6875 | 67 | 0.47 | 0.3125 |
| 11/6/2022 21:46 | 66.5375 | 64 | 3.96 | 2.5375 |
| 11/7/2022 17:40 | 65.53 | 66 | 0.71 | 0.47 |

The table shows that the system temperature sensor has exceptional accuracy when compared to a purchased sensor. The max error found was just under 4% which is well within the level of accuracy which we were hoping for.

**Humidity**

The sensor was placed outside at different times and compared to the humidity given by Weather.com. The test was performed like this because the commercially available sensor turned out to be incredibly slow and inaccurate. It was also done in this way because an outdoor environment is where the robot will be used.

| **Date/Time Recorded** | **System Humidity Sensor (%RH)** | **Website Listed Humidtiy (%RH)** | **Percent Error (%)** | **Difference (%RH)** |
| --- | --- | --- | --- | --- |
| 11/5/2022 23:30 | 82.3 | 84 | 2.02 | 1.7 |
| 11/6/2022 0:44 | 87.1 | 89 | 2.13 | 1.9 |
| 11/6/2022 1:23 | 93.1 | 95 | 2.00 | 1.9 |
| 11/6/2022 11:53 | 86.9 | 86 | 1.05 | 0.9 |
| 11/6/2022 15:06 | 83.2 | 84 | 0.95 | 0.8 |
| 11/6/2022 16:33 | 95.4 | 99 | 3.64 | 3.6 |
| 11/6/2022 21:46 | 92.8 | 96 | 3.33 | 3.2 |
| 11/7/2022 17:40 | 37.5 | 38 | 1.32 | 0.5 |

The table shows that the humidity sensor is well within our desired range for accuracy. The datasheet for the DHT22 sensor says the accuracy should be ±2 %RH. While this experiment found several points outside this range, it seems to mainly be at the higher humidity values where the accuracy decreases. However, even at these points, the accuracy is at a very acceptable level.

**Moisture Content**

The Moisture Probe was first tested using axial resistors so that a known value could be tested. The resistors were also checked on a multimeter so that the true value could be compared along with the listed value. This test provided limited results due to the availability of resistors to test. However, it is secondary to the full moisture content testing and was mostly used for troubleshooting.

| **Known Resistance (MΩ)** | **Multimeter Result (MΩ)** | **Probe Result (MΩ)** |
| --- | --- | --- |
| 1 | 1.02 | 1.02 |
| 2.2 | 2.23 | 2.23 |
| 4.7 | 4.83 | 4.82 |
| 10 | 9.88 | 9.78 |
| 47 | 51.0 | 46.0 |

The results above were all found to be accurate within 1% error except for the 47 MΩ test which had about 10% error. This is because it was found that the first two stages had greatly increased accuracy compared to the stages with larger resistors used in the voltage divider. After much research, it was found that Arduino analog read pins prefer a smaller input impedance which was unknown at the time of design. This could be fixed in future revisions by switching out the microcontroller used.

The system was used on blocks of wood with differing moisture contents and compared to the results given by a commercial wood moisture probe. Forcing a piece of wood to be a certain moisture content caused some problems with this experimentation. Leaving the wood pieces in a steamy room allowed values between 7% and 11% to be tested but the higher values could not be achieved this way. Instead, a piece of wood was soaked with water and tested at different times as the wood dried out and its moisture content fell. This allowed the higher values to be tested.

| **COTS Moisture Content Probe** | **System Moisture Content Probe** |
| --- | --- |
| 7 | 6.86 |
| 9 | 7.34 |
| 10 | 8.13 |
| 10 | 8.31 |
| 11 | 9.51 |
| 13 | 11.31 |
| 13 | 14.44 |
| 14 | 15.89 |
| 15 | 12.26 |
| 16 | 17.55 |
| 17 | 16.87 |
| 19 | 18.55 |
| 24 | 22.4 |
| 25 | 23.26 |
| 26 | 23.54 |

As can be seen from the table above, the actual moisture content value is not 100% accurate for the commercial probe. However, all values fall within ±3 of the expected percentage value. The collected data shows that the system is detecting changes within the moisture content of the plank. Based on the raw resistance values which were seen in the previous section. It is believed some of this error comes from the equation created to convert resistance to wood moisture percentage.

**Movement**

**The Height Constraint**

For the first test, we measured the height of the robot with a tape measure. This was done to ensure that the 16-inch height constraint was met. This experiment was conducted twice and the results are displayed below.

| **Trial Number** | **Height of Robot** |
| --- | --- |
| 1 | 15.1875 inches |
| 2 | 15.1875 inches |

The above results show that the robot is 15.1875 inches tall. That number is less than 16 inches, and thus the height constraint has been met.

**Motor Encoders**

The purpose of this test is to confirm that the motor encoders and Arduino software are correctly measuring the rotational speed of the motors. The motor encoders are connected to GPIO pins on the Arduino, and the software reads the frequency of the signals and converts them into a rotational speed that is displayed on the serial monitor. Those values were compared against a measurement obtained by using a digital tachometer placed in contact with the drive wheel. The values in the table below are in units of RPMs.

| **Robot's Reading** | **Tachometer Reading** | **Percent Error** |
| --- | --- | --- |
| 219.7 | 220 | 0.136% |
| 218.5 | 219.2 | 0.319% |
| 220.1 | 219.8 | 0.136% |
| 219.2 | 219.9 | 0.319% |
| 220 | 220.6 | 0.272% |

These results show that the Arduino program for the motor encoders is correctly measuring its speed.

**Linear Speed**

For this experiment, the team wanted to measure the robot’s linear speed. The desired speed is about 1 foot per second. To do this, we measured out 10 feet with a tape measure and then used a stopwatch to determine how quickly the robot covered that distance. This gives us the average velocity of the robot.

The measured values for average velocity were also compared to the instantaneous RPMs that were displayed in the Arduino Serial Monitor while the robot was in motion. Those values were recorded and are displayed below. The RPMs are also converted to feet/second for ease of comparison.

| **Travel Time** | **Average Speed(ft/s)** | **Instant. Speed(RPMs)** | **Instant. Speed(ft/s)** | **Percent Error** |
| --- | --- | --- | --- | --- |
| 9.75 | 1.03 | 138.7 | 1.09 | 5.505% |
| 9.68 | 1.04 | 135.8 | 1.07 | 3.21% |
| 9.71 | 1.03 | 137.8 | 1.088 | 5.331% |

These results show that the average speed of the Croomba is about 1 foot per second which is the desired value.

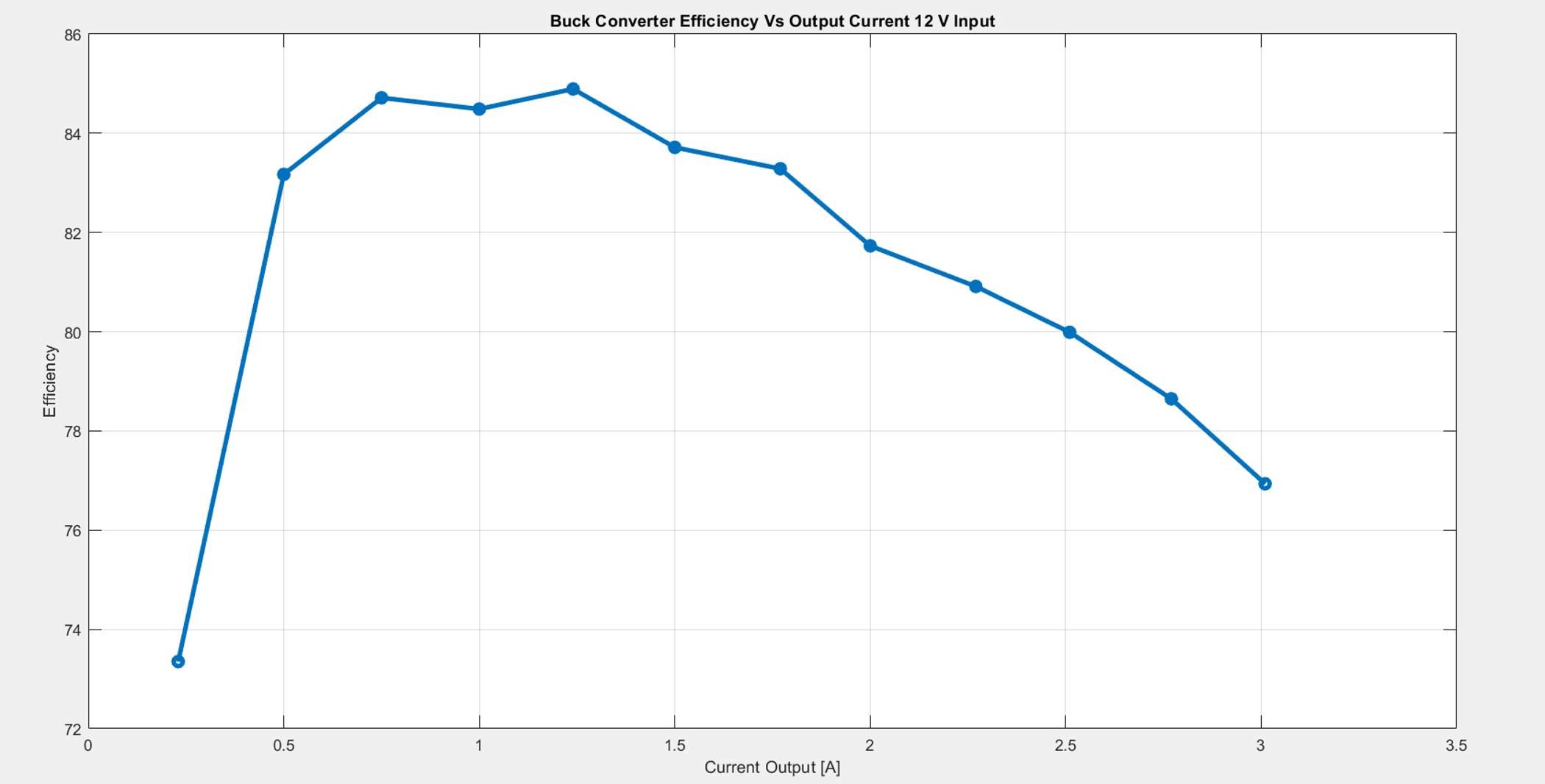
**30 Degree Incline**

Finally, a test was done to determine whether or not the Croomba could ascend a 30-degree inclined plane as described in the signoffs. To accomplish this, the team put together some pieces of plywood at a 30-degree angle and set the Croomba on a course to see if it could make its way up the hill. This test was conducted 3 times.

The Croomba was unable to go up the steep slope. The team believes this deviation is caused by a lack of traction on the treads as well as a high center of mass on the robot. That high center of mass pulls the robot back down the hill and the front of the bot begins to lose traction with the ramp. In short, the robot is too top and back heavy, and this leads to the robot wanting to flip onto its back like a turtle when ascending the incline.

However, the team conducted additional experiments by lowering the angle of the incline to find the maximum angle that the robot could traverse. All of the results are displayed in the table below.

| **Incline Angle** | **Croomba Ascension?** |
| --- | --- |
| 30 | No |
| 30 | No |
| 30 | No |
| 25 | No |
| 20 | No |
| 15 | Yes! |
| 15 | Yes! |



This figure was made by plotting the efficiency vs output current using an electronic load and lab test equipment. The efficiency of the converter is measured at various loads across the operating region here.

**Average Efficiency @12V: 81.39 %**