Operation

Operating the robot requires a 12 V power source with at least a 10 A discharge rate. The user will take the upper chassis off and connect the terminals of the power supply to the power switch. Once connected, flip the switch to ensure that power has been connected correctly. After connecting the power source, the user can then turn off the switch and reconnect the upper and lower chassis. If using a power source connected to AC mains, the user should ensure that the cord of the power source does not get caught.

After connecting the power supply and reconnecting the upper chassis, the user can turn on the robot and calibrate the line-following sensor arrays. To calibrate these sensors, hover each individual sensor over the white and black portions of the track to ensure that each sensor sees a high and low reflection level. It is important that this is done immediately after powering on the robot. After calibrating, set the robot down in the start square and wait for the icon to appear on the LCD screen, signaling that the Raspberry Pi has booted up. Once this icon has appeared, simply flip the start switch (the forwardmost switch) to begin playing the game.

Experimentation and Results

To first compare our final product to our original metrics of success, the metrics themselves must be revisited. These goals are listed in the table below.

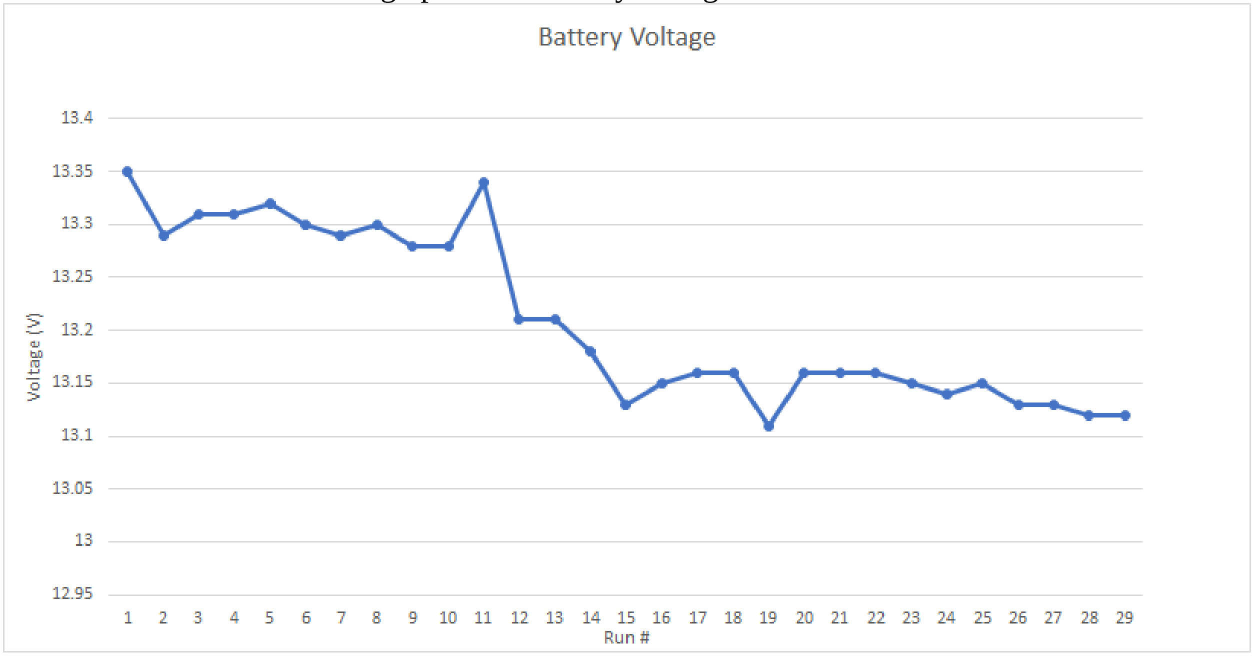
Measure of Success	**Target Value**
Size Specification	12"x12"x12"
Battery Life	1 hour @ 6 Amps of Current Draw
Bead Pick %	90%
Bead Pick Time	20 Seconds
Shot Accuracy	95%
Shot Time	20 Seconds
Net Detection	100%
Beads in All 3 Compartments	Pass/Fail
Marshmallow Moved	85%
Traversal Speed	2 ft/s
Run Time	<3:00
Run Points	40

The first metric, the Size Specification, is confirmed by simply measuring each dimension of the robot. The robot measures 11 7/8 " Long x 11 1/2" Wide x 11 3 /4" Tall , so it conforms to this spec.

To assess the success of the design as compared to the other metrics, full trial runs of the competition were simulated using a 1:1 replica of the game board. Different configurations (# of beads per tree, # of nets available, placement of marshmallow) were used to assess the performance of the robot through different game conditions. In total, 29 runs were attempted and completed to various degrees.

<u>Battery Life- Target 6 Amp Hours</u>

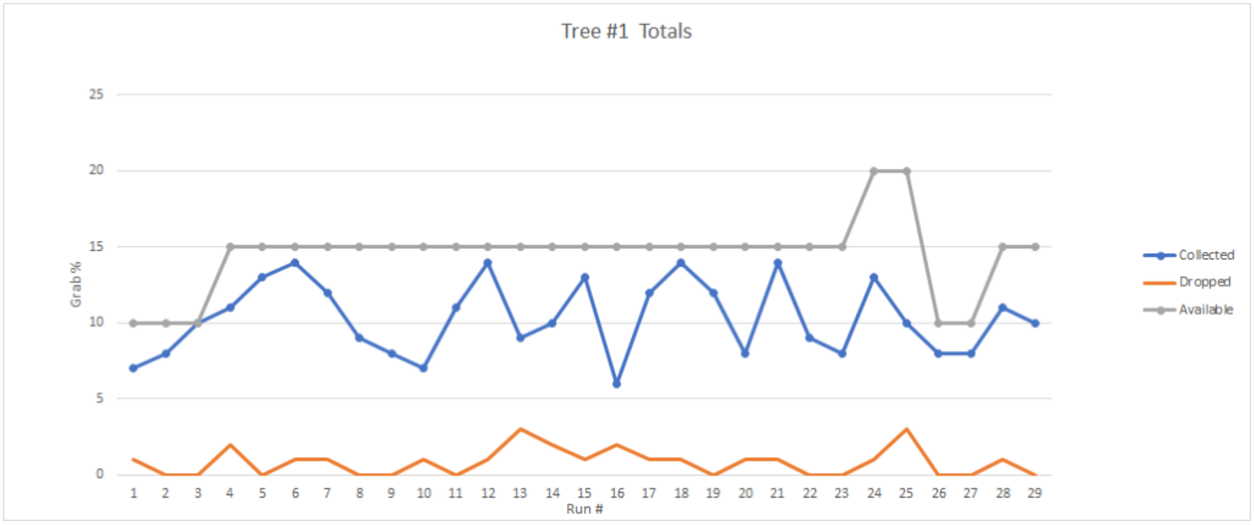
To test the battery performance and verify that it can support our system for an hour at 6 Amps of Current Draw (The expected Max Draw of the system) the battery voltage was measured before each run. A graph of the battery voltage after each run is shown below.

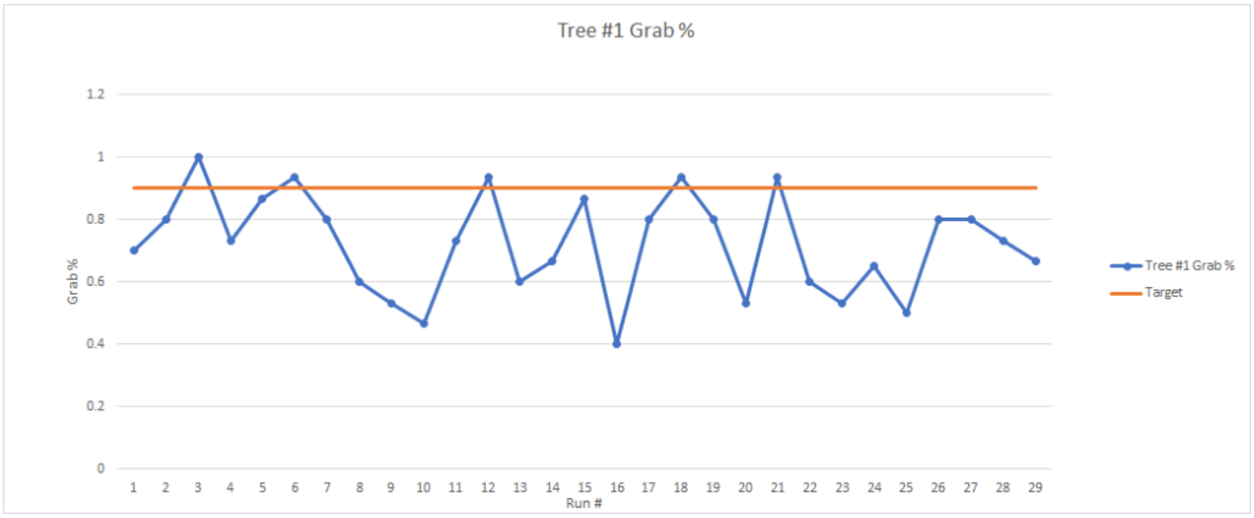


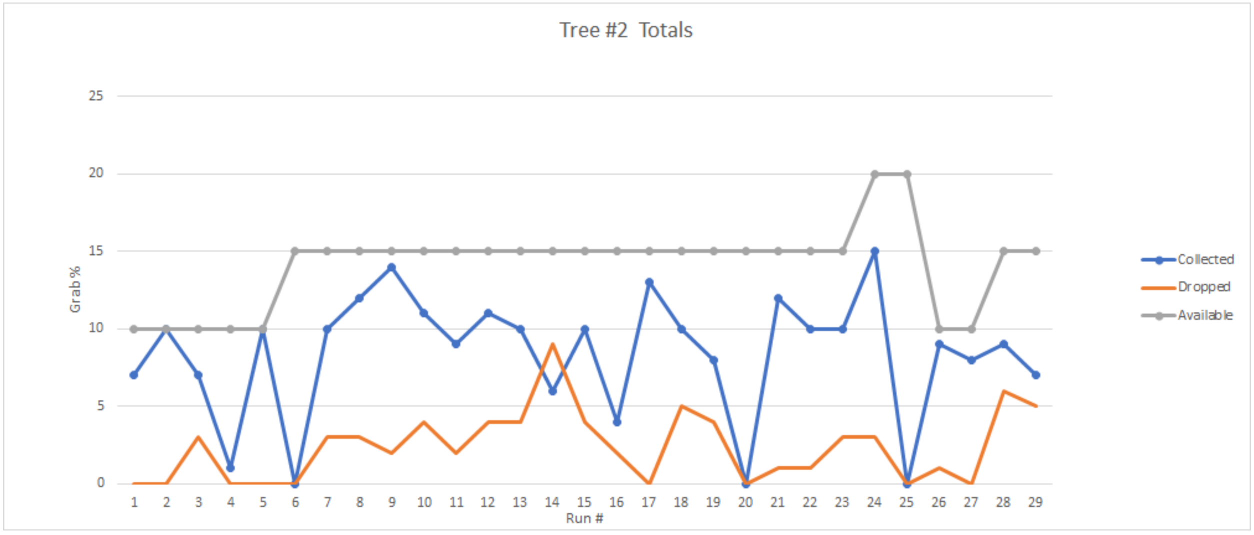
To prevent battery damage, the team determined that a measured battery voltage below 13 Volts indicated that the battery should be charged. The data above shows that this threshold was never crossed after the 29 Runs were conducted. At an average run time of 2:28 and an additional average of 0:45 for the Pi to Boot (3:13 of total time per run) over 29 runs conducted, the total powered on time of the system during testing was 29 X 3:13 or 1:33:00. After over 1.5 hours of runtime, the battery was still maintaining a voltage level of ~13.12 Volts and was still above the charging requirement. It can be concluded that the power system meets its battery life requirement.

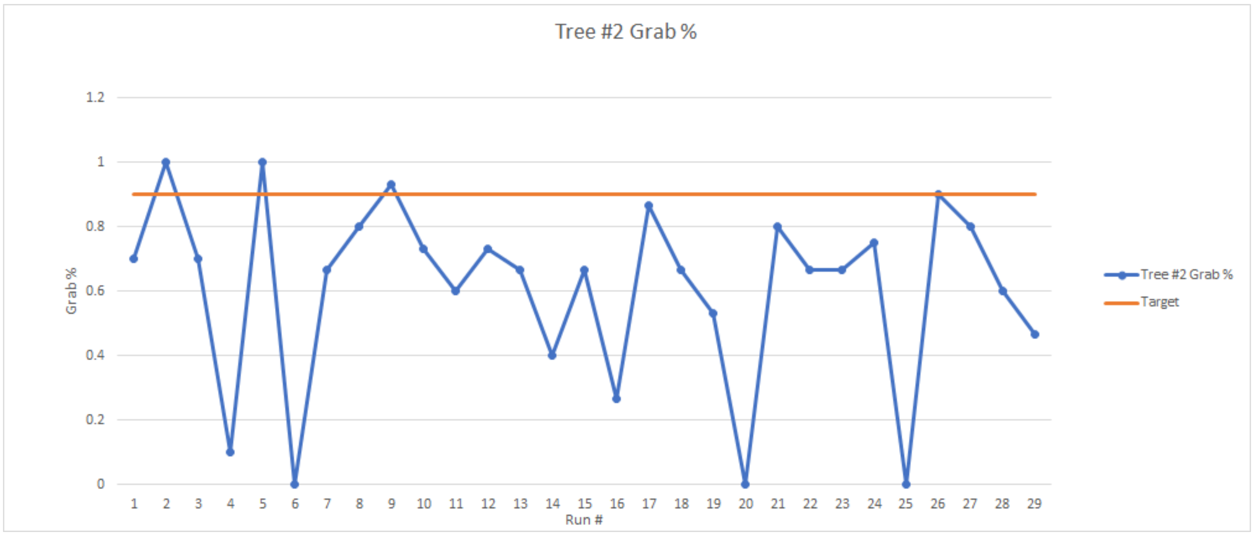
<u>Bead Picking – Targets: 90 % Picked in <20 Seconds</u>

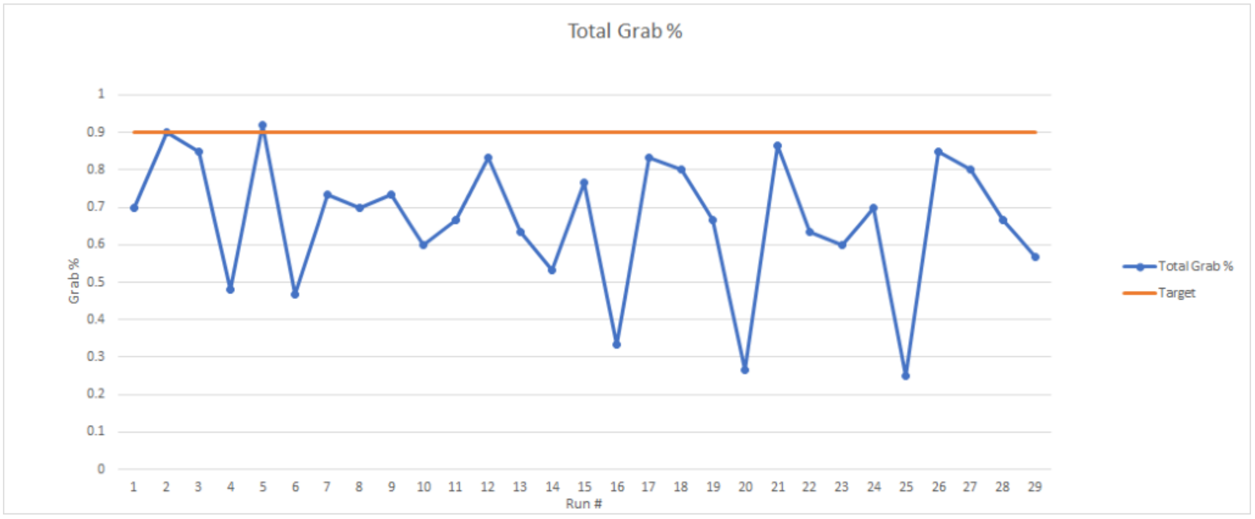
To measure the effectiveness of the robotic arm and its bead picking routine, the time of each pick, number of beads picked, number of beads dropped on the roadway, and number of beads available were recorded throughout each run. This data was split between each tree location, as each location was given its own arm routine to run. The pick % was also combined among both grabs per run. Graphs for each of these are shown below.


Tree #1 Totals
Grab %
Run #
Collected
Dropped
Available


Tree #1 Grab %
Grab %
Run #
Tree #1 Grab %
Target


Tree #2 Totals
Grab %
Run #
Collected
Dropped
Available





The data shows that the design fails to meet the original goal of a 90% pick rate. Additionally, the percentage tends to decrease as the number of beads on the tree increase. This demonstrates that the current design has an upper limit of beads that it can successfully pick without dropping a substantial amount onto the roadway. Also, these beads dropped onto the roadway tended to cause the robot to get stuck and fail to complete a full run of the course. These issues with bead picking can be attributed to two factors, the end effector and a misaligned 1st grab on tree #2. The end effector should have been longer and had more padding on the grippers to prevent them from not fully closing if two beads were grabbed side by side. This issue was common, and it would cause all of the other beads on that grab to be missed. The issue with the first grab at tree #2 was originally a misalignment issue that caused the arm to completely miss the beads. This meant that when the second grab did successfully reach the beads, there would be too many for a single grab, resulting in beads either being left on the tree or dropped onto the road. During the experimentation, the misalignment issue was adjusted and corrected. However, there were still issues with how this grab was placing the beads into the magazine. Instead of missing the beads altogether, this grab was now dropping some of them behind the robot instead of into it. Each time this happened, the robot was guaranteed to become stuck when attempting to return to the start of the track.

These issues with the bead grabbing system are the cause of many of the other performance issues of the robot. Instances of completely missed shots, getting stuck, and 0% of beads being grabbed from tree #2 were almost always caused by beads being dropped on the road. Either the robot would become stuck on these beads, or the Ultrasonic Sensors would mistake the beads on the road for a wall, causing the robot to stop short of its target.

Regarding the bead grab time, this was almost double the original goal of 20 seconds. Each bead pick was consistently 38 seconds with small deviations accounted for by imprecise timing. This is because of the additional grab at each tree. Originally only a single grab was anticipated, so an additional grab would be expected to double the grab time. However, other improvements to the robot's design, primarily the shooting time and traversal time, allowed these longer grabs to happen without exceeding the 3-minute time limit.

<u>Shooting – Targets: 95% Accuracy and < 20 Seconds</u>

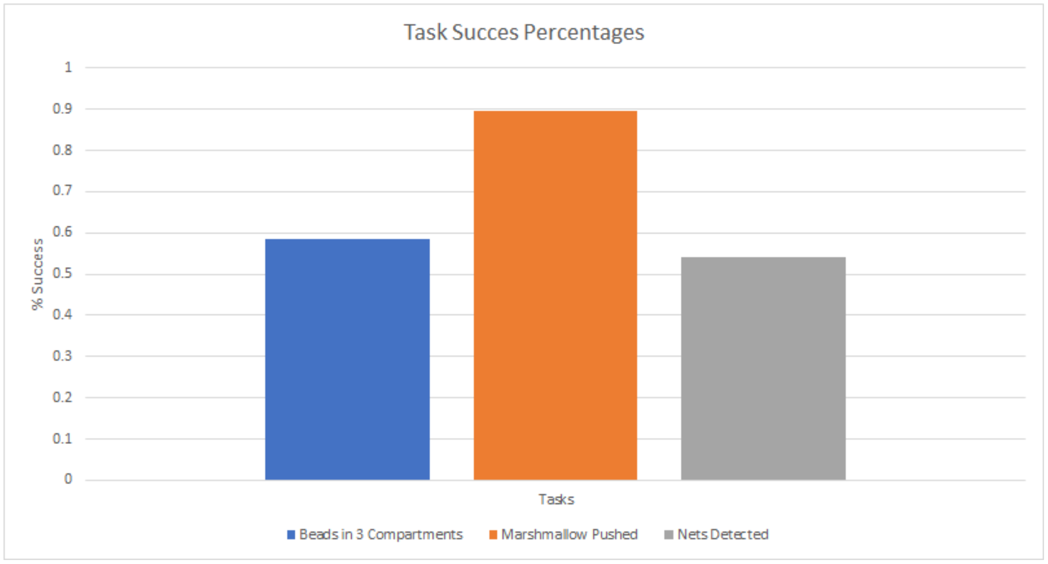
To measure the performance of the shooting mechanism, the number of beads fired, successful hits, and beads shot onto the roadway were tracked on each trial run. To match the specifications of the competition, beads that were not fully in the net, i.e., hanging halfway out, were counted as misses. Also, the disparity between the number of beads collected and beads fired is because some beads would be pulled off the tree and dropped into the robot chassis instead of the magazine.

Like the bead picking, shot accuracy failed to meet the original measure of success. However, the instances of 0% accuracy shown on the chart above can mostly be attributed to the robot failing to reach a net due to being stuck or lining up incorrectly due to bad sensor readings. As previously discussed, this was often caused by beads left on the track during the bead picking phase. Without these instances, the accuracy would be substantially better, but would often still fail to meet the original desired metric. The main cause of this would be too many beads fired at a single net. Beads would clump together and the shooting mech would not generate enough momentum to launch them fully into the net. This could possibly be alleviated by opening the magazine more slowly and spinning the motor up more before firing. Also, too many beads in a single shot is more likely when more beads are on the trees, which is why the accuracy is at its highest when 10 beads are on each tree.

Unlike bead picking, however, the shooting time was much lower than the original goal. Where the original metric was 20 seconds, the robot was able to stop, line up, fire, and drive again in under 8 seconds. As previously discussed, this shortened time allowed for more time to be invested in the bead grabs.

<u>Other Tasks: Targets: 100% Net Detection, 85% Marshmallow Moved, Beads Wanted in All 3 Compartments</u>

Other tasks that the robot was required to perform included detecting nets to shoot, moving the marshmallow, and distributing at least 1 bead into each compartment of the magazine. The percentages of each of these tasks is shown below.



Beads in All 3 Components

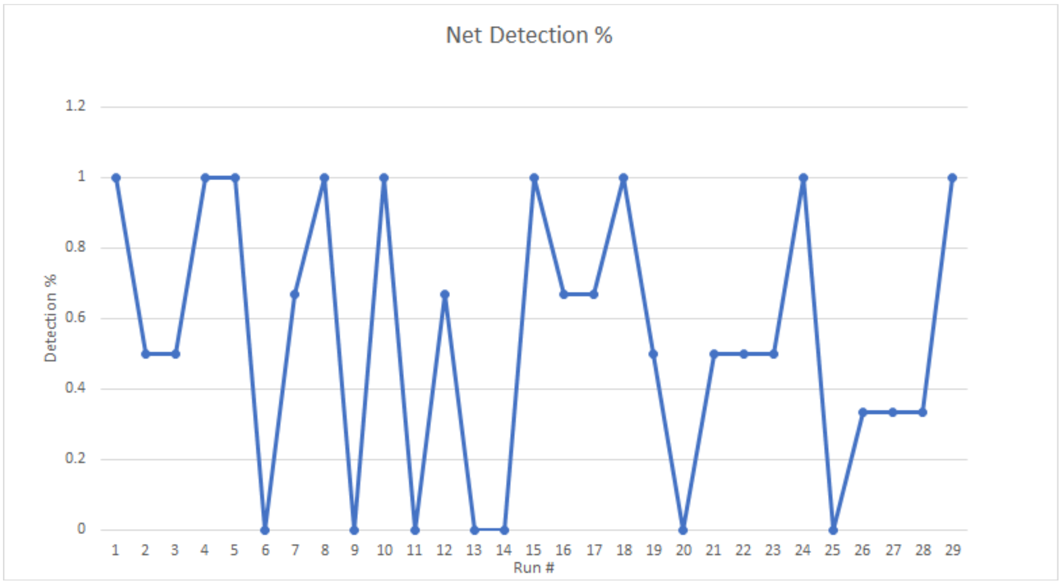
While it was originally desired to always fire beads into 3 nets regardless of their location, limitations, and optimizations closer to competition time required the robot to be tuned to instead only search 3 discrete locations for nets. If only 1 or 2 nets were identified instead of all 3, multiple compartments would be opened at a single net instead. Because of this, the emphasis on having beads in all 3 components was reduced in the final design. Despite this, the task was still accomplished almost 60% of the time. Also, many of these failures were either caused by the robot never reaching the second tree, which is where beads were pulled from to be placed in compartment 3 or by the 2nd tree's first grab being misaligned. Once the improvements were made to the second tree routine, the success rate of this task was 100% reliant on the robot making it to the second tree. If that was accomplished, beads were placed in all 3 compartments every time.

Marshmallow Pushed

The marshmallow was pushed off of the track 100% of the time that the robot successfully reached/lined up with the gap in the barricade. The only times the marshmallow did not get pushed out of the way were when beads on the track either caused the robot to become stuck or interfered with the ultrasonic sensors, tricking the robot into thinking the gap was in a different place. Despite these errors, the robot was still able to move the marshmallow approximately 90% of the time.

Net Detection

It was desired that nets be detected 100% of the time, but the minimum requirement would be that at least 1 net was detected, allowing for at least 1 accurate shot per run. In its current state, the detection rate of all nets is approximately 50%, but the robot is much more likely to detect at least 1 net. The table below shows that this is achieved on every run, as long as the robot is able to traverse the course. Often, it would be impossible to determine if nets were detected because the robot would never make it to the shooting stage of its program. However, if at least one shot was fired, the number of detected nets could be determined by examining the number of compartments that were opened during the first shot. 1 compartment = 3 nets detected, 2 = 2 nets, 3 = 1 net detected. Overall, net detection was fairly accurate provided that the batteries for the lighting system were charged enough. Low batteries are suspected to be the cause of the drop in accuracy later in testing. Further improvements to the system would allow the robot to fire at all net locations, as it is believed that the vision system would be robust enough to detect nets at any location, provided the robot can fire at the locations.



Full Runs – Targets: 40 Points and <3:00

The point totals of each test run and the total times were recorded as if these were live competition runs. The points across each run are shown below. Additionally, each run ended before the 3:00 time limit, meaning that this goal was met.

While the target was not always reached, these point losses were usually attributed to incomplete runs or a large number of point deductions from dropped beads or barricades hit. The data shows that achieving at least 40 points is possible and a solid baseline for a successful run. With further improvements to bead grabs, issues returning to the starting square and dropping beads (the two biggest causes of point loss) can be reduced dramatically.

Traversal Speed – Target: 2 ft/s without leaving line

The robot rarely travels a large distance without stopping while completing competition runs, making it difficult to analyze the straight-line speed of the bot during these experiments. Because of this, separate runs in which the robot would only drive were completed to determine its speed and driving capability. The robot was tested for both full runs of the course (~8.75 ft) which included the 90 degree turn and half runs of the course that only measured its speed on the longer straight section.

Type	**Distance (ft)**	**Time (seconds)**	**Speed (ft/s)**
Full Track with Turn	8.75	12.42	0.704508857
Full Track with Turn	8.75	13.02	0.672043011
Full Track with Turn	8.75	12.07	0.724937862
Full Track with Turn	8.75	12.12	0.721947195
Full Track with Turn	8.75	11.68	0.749143836
Half Track	6	7.15	0.839160839
Half Track	6	6.26	0.958466454
Half Track	6	6.41	0.936037441
Half Track	6	6.37	0.941915228
		Average	**0.805351191**

The data shows that the robot operates at an average speed of 0.805 feet per second. However, this could easily be increased by increasing the duty cycle of the PWM signals being sent to the drivetrain. Right now, that duty cycle is only 20 % – 25 % of the maximum. This was never increased because the current speed was adequate to meet the 3:00 time limit comfortably, and a faster speed was not necessary.

<u>**Power System – Target: Constant Output Voltages on Bucks and Buck/Boost Converters**</u>

The first element of the power system to be tested was the Buck/Boost Converter. The purpose of this device is to provide a constant output voltage of 12 Volts regardless of the input voltage. This was desired as it would normalize the power system's voltage level regardless of the varying battery voltage. To test this, a range of input voltages similar to the batteries operating voltages were supplied to the device, and then the output voltage was measured.

Input Voltage (V)	**Output Voltage (V)**
9.0	12.0
9.5	12.0
10.0	12.0
10.5	12.0
11.0	12.0
11.5	12.0
12.0	12.0
12.5	12.0
13.0	12.0
13.5	12.0

14.0	12.0

This data shows that the device works as intended and will be able to supply a steady output voltage even if the battery voltage is not constant. It is important to note, however, that the lower end of the test range is far below the recommended battery voltage level, and if the battery were to operate at such a low voltage, especially when connected to this device, the current draw would quickly deplete the battery and cause irreversible damage.

The second step in verifying the power system was to determine that the Buck Converters that serve as direct Power Supplies to the robot's electronics were capable of outputting a constant voltage given a constant input voltage of 12 Volts from the Buck/Boost Converter. Since the Buck/Boost has already been verified to output a constant 12 Volts if given any voltage in the range of 9 to 14 Volts, so by connecting these components to the Buck/Boost and measuring their outputs, their operation can be verified.

Device	**Test 1**	**Test 2**	**Test 3**	**Test 4**	**Test 5**	**Test 6**	**Test 7**	**Test 8**	**Test 9**	**Test 10**
Raspberry Pi Buck Converter (5 Volts)	5.0	5.0	4.9	5.0	5.1	5.0	5.1	4.9	5.0	4.9
Servo Shield Buck Converter (5.5 Volts)	5.5	5.4	5.4	5.5	5.5	5.5	5.5	5.4	5.5	5.5
Arduino Buck Converter (7.0 Volts)	7.1	7.1	7.0	7.1	7.0	7.1	7.0	7.1	7.1	7.1

This table shows that each device can maintain its voltage level with little variation. Each converter was tuned in such a way that small changes in voltage, either increases or decreases, would not exceed or fall below the connected electronics' operating voltage range. The Raspberry Pi can operate between 4.75 and 5.25 Volts, The Servo Shield can operate between 5.0 and 5.7 Volts, and the Arduinos can operate between 5.0 and 7.2 Volts (with a desired voltage closer to 7.0 Volts). Each of these specifications are met by the appropriate device.