Speed D8ers Report

Engineering 1282H

Spring 2018

Team D8

Justin Banke

Jack McGinness

Kelly Meaden

Colby Williams

R. Freuler Mon/Wed/Fri 10:20

Date of Submission: 04/23/18

Executive Summary

The Speed D8ers spent more than two months brainstorming, designing, building, coding, testing, and improving a robot prototype to be used in the inaugural Formula EH Grand Prix. Members of the Ohio State Research and Development (OSURED) team presented the need for a consistent and efficient pit assistant at the FEH Grand Prix on January 26, 2018, including the robot specifications and tasks to be completed. It was important to create an autonomous pit assistant to cut down on costs and safety concerns while also improving efficiency and drawing public attention to the event.

After brainstorming, the Speed D8ers designed a prototype with a wooden chassis, two IGWAN-motor-powered wheels and two unpowered wheels, and a forklift mechanism to pick up a wrench and a pronged mechanism to spin a fuel crank. Team builders constructed the robot over the course of the project while team programmers coded the Proteus microcontroller to have the robot navigate the course and complete each task. After analysis and testing, the final design included a wooden chassis, two IGWAN-powered wheels and two skids, a forklift mechanism to pick up and deposit the wrench and a flat, frictional mechanism to spin the fuel crank. The robot completed all primary tasks and 83% of the total tasks in its best trial during an individual competition on March 30, 2018. Although the IGWAN motors weakened before the final competition, the Speed D8ers' robot finished in 75th percentile at the final competition on April 7, 2018 after completing 84% of all tasks in its best run. Before production, the team planned to add a larger, more frictional surface to the fuel crank mechanism, while also increasing the speed of the total run and the performance and longevity of all motors.

Table of Contents

List of Figures

List of Tables

1. Introduction

Since the invention of the automobile, people from across the world have been fascinated by racing. With dozens of cars competing for the top spot, a precise and efficient pit crew is essential to an individual car's success. The difference between winning and losing can often be traced back to quick pit stops. Pit crews have become even more prominent in recent months, as NASCAR announced that over-the-wall pit crews will be reduced to five members instead of six during the 2018 season [1]. Even more drastic, the Formula EH (FEH) Grand Prix plans to utilize unmanned vehicles in the pit to automate tasks in its inaugural race. Not only will unmanned vehicles save money and eliminate safety concerns, but FEH executives also expect increased efficiency and public attention as a result [2]. FEH executives set up a competition to select a prototype from 63 entries to be used in the Formula EH Grand Prix. The Speed D8ers designed, built, and tested a prototype for the autonomous pit assistant in the Formula EH Grand Prix.

The project officially began on January 26, 2018, when FEH executives presented the problem to engineers. Justin Banke, Jack McGinness, Kelly Meaden, and Colby Williams formed the Speed D8ers company on January 31. During the subsequent two months, team members collaborated to create an autonomous pit assistant for the Grand Prix utilizing programming, computer-aided design (CAD), and other engineering skills acquired in previous coursework and experience. The Speed D8ers' robot was tested individually on March 30, and it competed head-to-head against sixty-two other vehicles on April 7, 2018. The company presented its work to the Ohio State Research and Development (OSURED) team on April 18, which selected the prototype to be used at the inaugural Formula EH Grand Prix [2].

The remainder of this report details the Speed D8ers' project from start to finish. The next section, Preliminary Concepts, presents the preliminary ideas and mockup that resulted from group brainstorming sessions. Section 3, Analysis, Testing, and Refinements, details the decisions made leading to the final design, as a result of analysis, calculations, and testing. Section 4, Individual Competition, describes the individual competition and the robot's performance. The final design is depicted in section 5, Final Design, with details of project management and specific features of the robot. Section 6, Final Competition, reports on the final competition and the robot's performance. Section 7, Summary and Conclusions, summarizes the project and provides suggestions for future work and possible enhancements. Finally, section 8 lists the documents referenced throughout the project and this report.

2. Preliminary Concepts

Before forming the company, the engineers individually brainstormed strategies to navigate the course, designs for the robot's chassis and drivetrain, and mechanisms to complete several tasks. Once the Speed D8ers formed, the team members combined their ideas and deliberated further to come up with preliminary ideas based on the project's requirements and constraints. Together, the team conceptualized three full-vehicle designs and then created physical and virtual mockups of the best design.

2.1 Project Requirements and Constraints

The robot was required to complete a number of tasks quickly and precisely to successfully aid pit crews at the Formula EH Grand Prix. OSURED constructed a scale model of the pit and garage area, which was available for use by the company during specified business hours. An overhead view of one section of the course is shown in Figure 1 below. The garage was elevated above the pit, and it was accessible by a grass ramp and a concrete ramp [2].

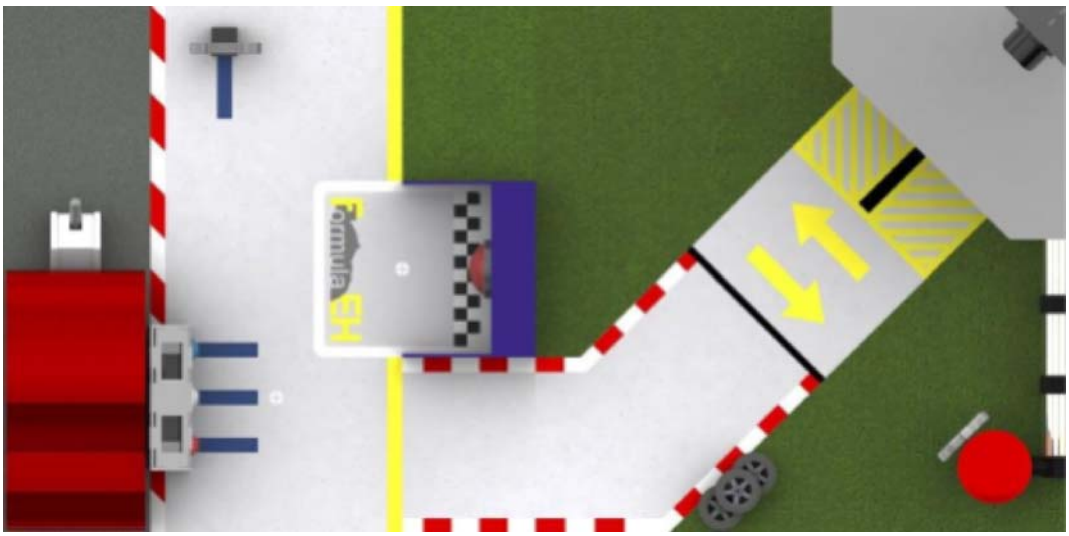


Figure 1: Overhead view of the pit area and garage, as provided by OSURED [2].

According to the project guidelines provided by the Formula EH executives, the following tasks had to be completed by the pit assistant in less than two minutes. First, the robot had to start when the light in the starting area beneath the robot was illuminated, thus clearing it to begin operations. From that point, the pit assistant had two minutes to complete the specified tasks before returning to the starting area and pressing the final charging button. Race cars often enter the pit to refuel with either 98 octane gasoline or nitromethane, so one primary task of the robot was to turn the fuel crank in the direction of the corresponding fuel type. During the pit

stop, the necessary fuel type for the race car was transmitted to the pit assistant wirelessly, and the robot was responsible for turning the crank 180 degrees clockwise for 98 octane gasoline or 180 degrees counterclockwise for nitromethane gasoline. The pit assistant was also responsible for clearing the area of any maintenance equipment. Thus, the robot had to pick up a wrench from the pit and transport it to the garage. Before the race car could leave, the pit assistant had to release it from the car jack used when changing tires. The final task was to perform a diagnostic test on the race car to test either its telemetry sensors, which allow the pit crew to monitor engine speed and provide a live video of the driver, or its electronic control unit (ECU), which controls the vehicle's numerous sensors. The race car only needed one of these tests during each pit stop, so the pit assistant had to scan a light in front of the control panel to determine which test to conduct before pressing the corresponding button. Once these tasks were completed, the robot had to return to the starting area to recharge [2].

The Speed D8ers considered each of the required tasks when developing a design for their prototype. Additional consideration was given to the Robot Positioning System (RPS), which wirelessly provided information regarding the robot's position in specified areas on the course. RPS was active in the pit during the entirety of each run, and communication could be activated on the upper level if an additional white button on the control panel was held for five seconds. The circular area around the garage, with a radius of 20", lacked a signal regardless, as indicated by the experimental guidelines [2].

The robot also had to meet several specifications, as indicated by the experimental guidelines. First, it had to fit within a 9" x 9" footprint before starting, and it could be no more than 12" tall. Its programmable Proteus microcontroller, which was lent to the company by

OSURED, could not be used as a sensor or structural support, and the only allowable adhesive was Velcro. The robot had to be fully autonomous, and a QR code had to be mounted 9” above the course surface for proper communication with the control systems. Finally, the budget for the prototype was $160 in addition to a set of basic sensors and the Proteus. Most parts, including various motors, chassis materials, and drivetrains, could be ordered from the FEH Company Store [2].

2.2 Brainstorming and Idea Screening

Individually, each team member came up with three course strategies, three chassis and drivetrain combinations, and three mechanisms for completing each task. After forming the company on January 31, team members discussed and compared ideas, and generated some new ideas as well. Next, the company screened the top ideas for each chassis and drivetrain combination and for several mechanisms. Each idea was rated on its ability to meet certain criteria, such as speed, cost, and durability. Screening matrices compared each idea to an average reference, with a “+” representing better performance than the reference, a “-” representing worse performance than the reference, or a “0” suggesting similar performance to the reference. To decide which concepts to proceed with, the company totaled the “+”s, “-”s, and “0”s to form a net score of each idea. This was completed with at least four ideas for the chassis and drivetrain, the car jack mechanism, the fuel crank mechanism, and the wrench mechanism. Each screening matrix can be found in Tables A1-A4 in Appendix A, and is discussed in detail in sections 2.2.2 through 2.2.6.

2.2.1 Course Strategy

The discussion regarding course strategy centered around three main ideas: starting with the buttons and moving clockwise around the course, starting with the wrench and moving counterclockwise around the course, or using two separate chassis connected by a wire to complete two tasks at a time. Additional deliberation focused on the following ideas: pressing the white button to activate RPS, depositing the wrench immediately after picking it up, and the possibility of pressing the final button from the upper level. Ultimately, the team settled on using only one chassis to limit complexity and cost. The top strategy suggested navigation in the following order: pressing the white RPS button followed by the red or blue button on the control panel, toggling the car jack, picking up the wrench, depositing the wrench, rotating the fuel crank, and pressing the final recharging button from the starting area on the lower level. This strategy is depicted in Figure 2 below. The second-ranked strategy, which is depicted in Figure A1 in Appendix A was similar, but it started with the wrench, moved counterclockwise, and finished by pressing the final button from the upper level.

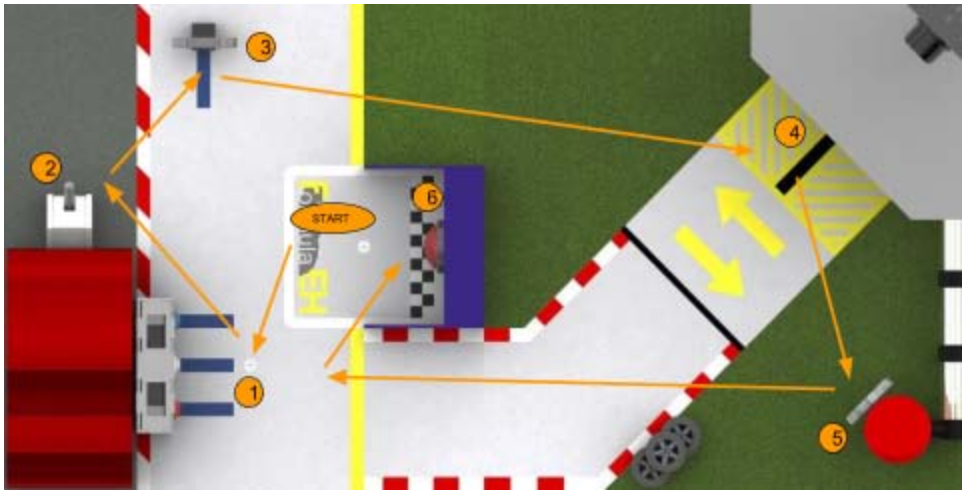


Figure 2: Top-ranked course strategy.

2.2.2 Chassis and Drivetrain

The discussion regarding the chassis and drivetrain centered around wheel orientation. The team came up with five chassis and drivetrain options: four powered wheels, two powered and two unpowered wheels, a circular layout with OMNI wheels perpendicular to the turning direction, two powered wheels and a chain attached to two unpowered wheels, or two robots each with a triangular frame. The five ideas were screened, as shown in Table A1 in Appendix A. As a result of the screening, the team chose to proceed with four powered wheels, two powered wheels and two unpowered wheels, and a circular wheel layout.

2.2.3 Car Jack Mechanism

The team came up with four ideas for the car jack mechanism. A sloped front could run into the jack or a lever could lift it. Also, team could construct an arm to clamp onto and lift the jack, or they could place a rod on a gear that could turn upwards to toggle the jack. Since the jack could be toggled with relative ease, the team focused on simplicity when screening the ideas. The screening matrix is shown in Table A2 in Appendix A. As a result of the screening, the team proceeded with the sloped front and lever ideas.

2.2.4 Buttons

The team brainstormed four ideas for pressing buttons on the control panel. The team considered using a narrow part on the front of the robot to press buttons, or adding a plate that extended out of the robot. The team also considered attaching three rods to the robot, and sliding

two of them forward to press the white RPS button and the specified test button, or simply running into the buttons with the front of the robot.

The team also brainstormed five ideas for pressing the final button. The team considered driving into the button with the chassis, driving into it with a slanted sled, pressing the button with mechanisms used for other tasks such as a rod or a gear, constructing an arm to slap the button from the upper level, or adding a plate that extends from the robot to press the button. Since the ideas considered primarily used mechanisms from other tasks, the ideas were not screened using a matrix.

2.2.5 Fuel Crank

The team came up with four mechanisms for rotating the fuel crank. Team members considered using a rod on a rotating gear that would go into the crank and spin. Using a rotating disk that would align with the crank and spin with it was also considered, or constructing an arm that could move up and down to spin the crank. A final idea was placing two rods on opposite sides of the gear to rotate the crank. The screening matrix for these ideas is shown in Table A3 in Appendix A. The team proceeded with the rod on a gear and the rotating disk after screening the ideas.

2.2.6 Wrench

The Speed D8ers brainstormed four mechanisms for controlling and depositing the wrench. The company considered building an arm that could slide through the hole of the wrench and turn or lift to pick it up, or adding a forklift to the front of the robot to carry the wrench. Team members also discussed the possibility of using a rod with a stopper that could enter the

hole and drag the wrench to the garage. A final idea was constructing an arm with clamping ability to grab the wrench. These ideas were screened using the matrix shown in Table A4 in Appendix A. The team proceeded with all of the ideas except for the arm with a clamp due to its cost and inefficiency.

2.3 Preliminary Ideas

Next, the Speed D8ers took the top ideas produced from each of the screening matrices and produced three combinations for a full robot layout. These combinations became the company's preliminary ideas for the robot layout. In their basic form, the combinations are shown in Table 1 below. Each combination is discussed in more detail in sections 2.3.1 through 2.3.3.

Table 1: Combinations of chassis and drivetrains, and mechanisms for the preliminary ideas.

Components	Idea One	Idea Two	Idea Three
Chassis and Drivetrain	aluminum, 2 motored wheels	PVC, 2 motored wheels + chain	wood, 4 motored wheels
Car Jack	Sloped Front	Lever	Lever
Fuel Crank	Rotating Disk	Rod on Gear	Two Rods on Gear
Wrench	Rod with Stopper	Arm In Hole	Forklift

The team used a concept scoring matrix, as shown in Table A5 in Appendix A, to rate each idea on its ability to meet specified criteria. In the concept scoring matrix, each combination was given a score from 1-5 for each criteria and compared to a reference concept, which received an average score of 3. Each criteria was weighted, so the importance of each requirement was reflected in each concept's score. For example, the company determined that maneuverability

was the most important criteria and should be 25% of the total score, while less prominent criteria were a smaller percentage. The weighted score for each criteria was calculated by multiplying its rating (on a scale from 1-5) by the criteria's percentage of the total score. Then, the team added the weighted scores to give each idea a total score. Finally, team members sketched each robot concept, and weighed the pros and cons of each. Then, the team produced a physical and CAD mockup of their top idea while considering the practicality of the preliminary concepts and making adjustments.

2.3.1 Preliminary Idea One

The first preliminary idea is depicted in Figure B1 in Appendix B. It consisted of a rectangular aluminum chassis with four wheels attached. The front two wheels were powered, giving the robot front wheel drive and limiting the cost associated with additional motors. In this design, half of the front face was a sloped ramp, which could lift the car jack by running into it. The other half of the front face consisted of a disk connected to a motor shaft so it could spin. When pressed up against the fuel crank, the motor could rotate the disk, thereby spinning the crank. On the top of the robot was a rod attached to a servo motor. The robot could drive so that the rod went through the hole of the wrench. The servo could then rotate upwards, lifting the wrench off of its stand. The stopper attached to the rod could catch the wrench and prevent it from getting too close the robot. The servo could rotate the rod downward to deposit the wrench. All of the buttons could be pressed by driving into them with either the ramp or the rod.

2.3.2 Preliminary Idea Two

Figures B2 and B3 in Appendix B illustrate the second preliminary concept. The chassis consisted of a PVC plate attached to four wheels. The front two wheels were powered by motors and the back two wheels were linked to the front two wheels via a sprocket and a chain. The car jack could be toggled using a servo motor attached to an arm that could lift the jack. The same mechanism was used for both the fuel crank and wrench. A DC motor was attached to a rod with two bends. The rod could slide into the holes of the crank and spin the appropriate direction, as measured by the motor encoder. This mechanism could also lift the wrench. The robot would drive so that the arm slides through the hole of the wrench. The arm could then turn, raising and holding the wrench until it was dispensed.

2.3.3 Preliminary Idea Three

Figure B4 in Appendix B represents the third preliminary idea. A square, wooden chassis was supported by four wheels, each one powered by its own motor and attached to the underside of the chassis. On the front of the chassis was a mechanism with the ability to function as a forklift that could spin in addition to moving vertically. This mechanism could complete all of the tasks on the course. The two prongs of the mechanism could be inserted into the holes of the wrench. The forklift could then be raised to lift the wrench off of its stand and lowered to deposit it in the garage. The prongs could also be raised to the appropriate height in order to press the buttons. Finally, the height of the forklift could be adjusted so that it fit into the fuel crank. The motor could then rotate 180 degrees in either direction to turn the fuel crank.

2.4 Mockup

After reviewing the concept scoring matrix, shown in Table A5 in Appendix A, the team initially decided to proceed with Preliminary Idea One, which is described in section 2.3.1. However, after further discussion of course measurements, team members decided to proceed with a single mechanism to complete all three tasks, as detailed in Preliminary Idea Three. Thus, the mock-up was a combination of the two preliminary concepts. Both a SolidWorks mockup and a physical mockup were constructed. The physical mockup was created using cardboard, paper clips, and tape. A picture is shown in Figure 3 below. It represented a rectangular aluminum chassis folded up on all four sides, four wheels (powered in the front), two motors, and the Proteus. The mechanism added to the front was able to both spin and slide vertically. It represented a rack and pinion connected to two servo motors. A plastic disk was also added to the back as an additional means of pressing the buttons.

Figure 3: Physical mockup.

A SolidWorks model was then constructed to represent the mockup digitally. Measurements were made to ensure that the SolidWorks design resembled the physical mockup closely. The SolidWorks model is shown in Figure 4 below.

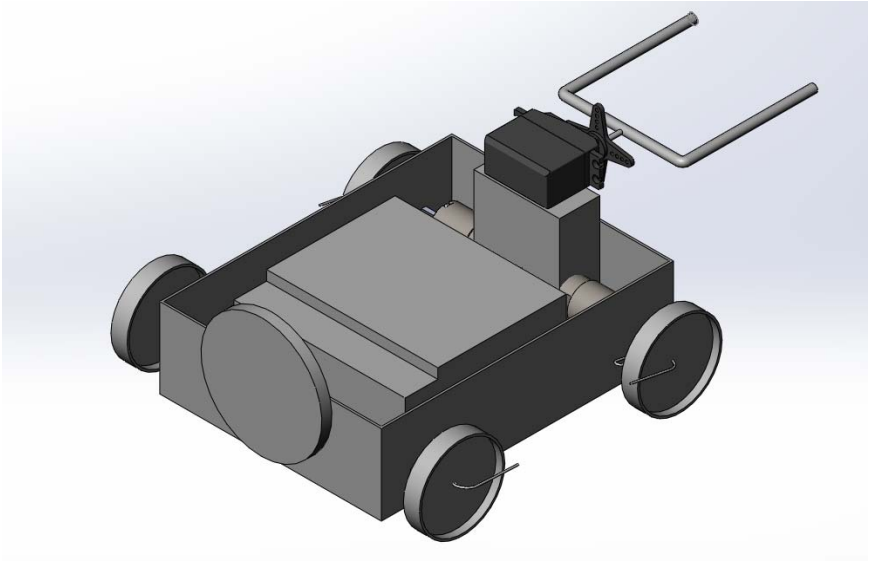


Figure 4: SolidWorks mockup.

2.5 Preliminary Code

While brainstorming ideas for the code, several different decisions were discussed. Most importantly, it was decided that a mixture of RPS and sleep statements would be used. The RPS would be utilized when the robot's position was crucial, as well as generally to make sure the robot was in the correct position. The sleep statements would be used to get the robot the majority of the distance to these positions, as the RPS would be slow. Sleep statements were decided on over encoders due to uncertainty over skidding at higher speeds. After some deliberation, a preliminary algorithm was made, shown in Appendix B.

3. Analysis, Testing, and Refinements

Throughout the duration of the project, the Speed D8ers analyzed, tested, and refined the robot. Team members took measurements of the course, calculated the torque and speed required for the motor, and explored color sensing and filtering, line following and shaft encoding, and RPS navigation. The company modified the prototype's design and code as a result of testing on the course provided by OSURED. The robot completed four performance tests and participated in an individual competition before the final product competed head-to-head in the final competition on April 7, 2018.

3.1 Course Measurements

Initially, the dimensions of many important course elements were measured. These findings directly influenced the design of the robot.

Table 2: Measured dimensions of important course components

Course Component	Dimension
Distance between centers of holes of the wrench	3.5"
Distance between centers of white button and red/blue button	3.25"
Crank diameter (excluding spindles)	3"
Diameter of holes of wrench	0.875"
Height of buttons off of ground	1.9375"
Height of center of wrench holes off of ground	2.0625"
Height of center of fuel crank off of ground	3.0625"

The close proximity of the first three dimensions listed in Table 2 on the previous page induced the idea of an all-in-one mechanism consisting of two prongs spaced 3.25” apart. The prongs would fit through the holes of the wrench. They would also be used to simultaneously press the white button and the red or blue button, depending on the color of the floor light. Finally, the prongs would be spaced to that they would fit on the outside of the crank and spin it using the spindles. Importantly, the buttons, wrench, and fuel crank were measured to have different heights. The disparity in heights required a method of raising the prongs. It was decided that the prongs should be able to spin in order to turn the crank, and should be able to translate vertically to lift the wrench and to raise to the fuel crank’s height. Thus, the team decided to use a rack and pinion mechanism. This idea was used in the team’s mock-up, which is described in section 2.4.

However, the complexity of the all-in-one mechanism, combined with the difficulty in buying a rack and pinion from an external source, led to a change in design. Two separate prong mechanisms were created, one for the buttons and wrench and one for the fuel crank. The button and wrench mechanism was the appropriate height and could lift upwards. The fuel crank mechanism was placed higher on the robot and had 3D printed prongs that rotated.

3.2 Drivetrain Calculations

To determine which motors were capable of successfully driving the robot through the course, calculations were performed. First, the speed necessary for the robot to complete the appropriate tasks in less than two minutes was determined. The expected travel distance was found by adding the distance of each segment of the course navigation strategy. Table C1 in

Appendix C lists the distance of each segment, totaling 193”. Next, the expected time needed to complete each action – excluding driving – was summed, as presented in Table C2 in Appendix C. This totaled to 26 seconds. After adding 9 seconds for buffer, it was determined that 35 seconds of the 2 minute round would be needed for tasks other than driving. This left 85 seconds to cover the 193 inches necessary to reach all of the tasks. The minimum linear speed of the robot was then calculated to be 2.3 inches per second using Equation 1 below. This calculation can be found in Appendix D.

$$v = \frac{distance}{time} \tag{1}$$

The respective minimum angular speed of the motor using a wheel of 1 inch radius was calculated to be 22 revolutions per minute using Equation 2, shown below. This calculation is shown in Appendix D.

$$\omega = \frac{v}{2\Pi r} \tag{2}$$

Next, the minimum torque provided by the motors to drive the robot up the ramp was calculated. By summing up the weight of each proposed component of the robot, an estimated weight of the robot was calculated. As demonstrated in Table C2 of Appendix C, the approximate weight of the robot was found to be 1059 g. This was converted to 37.36 ounces using Equation 3 below. This calculation can be found in Appendix D.

$$ounces = grams \cdot \frac{1\ oz.}{28.35\ g} \tag{3}$$

Figure C1 in Appendix C illustrates the forces acting on the robot as it drives up the ramp. The internal friction of the robot was assumed to be 8 oz. Using the measured height of the ramp, 3.125”, and the measured length of the ramp, 9”, the incline was found to be 19.15°. In

order for the motor to successfully propel the robot up the ramp, the force delivered by the motors had to at least equal the counter forces parallel to the ramp, weight and friction. Equation 4 below was used to calculate the required force of the motors. By the calculation shown in Appendix D, the force was found to be 20.26 ounces.

$$F_{motors} = F_{friction} + W sin(19.15) \tag{4}$$

Equation 5, found below, was used to calculate the torque necessary to deliver 20.26 oz of force using a 1" diameter wheel. The necessary torque was 20.26 ounce-inches, as found with the calculation shown in Appendix D.

$$\tau_{motors} = F_{motors} \times r_{wheel} \tag{5}$$

Because two motors would be used to drive the robot, this torque was halved; each motor would have to exert a minimum torque of 10.13 oz-in. Six motors were offered to power the robot: IGWAN, Acroname, VEX 393, GMH-34, FUTABA Servo, FITEC Servo. The minimum specifications of the motor for the robot – a speed of 22.0 revolutions per minute and a torque of 10.13 oz-in – were plotted on a Torque-Speed Curve for the six motors. This is illustrated in Figure C2 in Appendix C. Because the point lays below the torque-speed curves of all six motors, a pair of any of the motors would be capable of driving the robot up the ramp and to all of the tasks in less than two minutes. However, Igwan motors were chosen as the preferred motor due to its reliability, built-in encoders, and efficient attachment options.

3.3 Explorations

Three explorations were conducted to further the general knowledge of the team and provide potential methods to improving the success of the robot.

3.3.1 Exploration 1

The first exploration investigated the use of motors and sensors in the robot competition. A green filter paper was placed over a CdS cell, which is a small sensor used to detect the color of emitted light, to experiment in differentiating between a blue light, a red light, and the floor of the course. The CdS cell voltage reading for these three surfaces was recorded. The Proteus was then programmed so that a servo motor would be in position 0 when the CdS cell was in full light, position 180 when the CdS cell was in complete darkness, and any corresponding position in between. The Proteus was then wired to a Crayola Bot and programmed to navigate the robot through a simple maze using four bump switches.

The green filter proved to be unhelpful in distinguishing the red and blue lights; the values were in close proximity since green is close in color to both red and green. This result suggested the use of a blue or red filter, which would differentiate the readings better. The servo motor successfully represented the intensity of light received by the CdS cell and demonstrated its ability to hold a specific position. The Crayola Bot successfully navigated the course. The IGWAN motors smoothly translated the robot and the bump switches were essential in order to turn after hitting each wall. Minor issues were faced when the robot backed up after hitting a wall. Occasionally, one bump switch on a side would be pressed, but the robot continued turning

into the wall in a way such that the second bump switch on that side would never be pressed. Consequently, the robot would continue to be stuck on the wall indefinitely, waiting for the second bump switch to be pressed. This was resolved by editing the code. In this situation, the wheel opposite of the pressed bump switch was reversed.

Ultimately, the team decided to use a CdS cell, taped underneath the chassis, to read the lights on the course. No filter was used, as the CdS cell adequately portrayed the difference in voltage between red and blue light. The Speed D8ers opted against using bump sensors on the robot, as straightening into the wall was adequate while testing.

3.3.2 Exploration 2

The second exploration focused on line following and encoder-based navigation. Initially, the readings of an optosensor on a black line and on the background around it were recorded. The Proteus was wired to a Crayola Bot and programmed to follow a straight line using the optosensor. This code was enhanced to allow following of a line with curves. Calculations were also made to convert shaft encoder counts to inches travelled by the robot. Equation 6, shown below, was used to determine that there were 40.489 encoder units per inch that the robot travels, given that there were 318 encoder units per rotation and the wheel diameter was 2.5”. This calculation is shown in Appendix D.

$$\frac{encoder\ units}{1\ inch} = \frac{encoder\ units}{rotations} \cdot \frac{rotations}{\Pi d\ inches} \tag{6}$$

The Proteus was then programmed to make the Crayola Bot travel 6 inches at three different motor power percents: 25%, 40%, and 60%. The experimental encoder values for the left and

right motors were recorded after each trial. Finally, the Proteus was coded so the Crayola Bot would drive a specific sequence.

The optosensor was successfully used to follow the straight and curved lines. However, it was most successful following a straight line and at lower motor percents. The success demonstrated the potential benefit of following the lines on the robot course. For all three trials of travelling six inches, the experimental encoder values were different than the theoretical and not equal between the left and right motors. Table 3 below describes the experimental encoder values of the right and left motor shafts when set to travel 243 encoder units, for a theoretical distance of 6 inches.

Table 3: Experimental shaft encoder data for intending to travel 6 inches

Motor Percent	Experimental Distance Travelled (inches)	Experimental LE Counts Travelled	Experimental RE Counts Travelled
25%	5 ⅞”	256	250
40%	6 5/16”	276	270
60%	7 ⅛”	301	299

This was explained by the motor shafts continuing to turn after power was ceased. The discrepancy increased as motor power increased, revealing the greater accuracy presented by travelling at slower speeds. Additionally, the disparity between the right and left motors illustrated the slight inaccuracy of encoder based navigation.

While the Crayola Bot was able to perform the specific navigation sequence with little error, the team elected to primarily rely on time-based navigation rather than shaft encoding. Furthermore, the robot was able to press the buttons and pick up and deposit the wrench without

following lines, so the team determined that line following was not necessary to complete the required tasks. Thus, the robot did not utilize optosensors.

3.3.3 Exploration 3

The final exploration explored the Robot Positioning System (RPS) and data logging and their ability to enhance the success of the robot. A QR code was placed on the top of a Crayola Bot in order to be read by the RPS above the robot course. The RPS x and y coordinates sent to the Proteus were recorded at four specific locations on the robot course. The Proteus was then wired to the Crayola Bot and programmed to use RPS to check and adjust the robot's position after travelling measured distances using encoders. The driving motors adjusted the robot until its x position, y position, and heading were all within 1 unit of the intended position. Next, the position of the robot was logged on the microSD card every 10 milliseconds during an RPS enhanced sequence on the course. These positions were then plotted on a picture of the robot course using MATLAB, as shown in Figure C3 in Appendix C.

RPS proved to be very helpful in correcting encoder based movements. MATLAB properly used to map out the movement of the robot over an image of the robot course. This system of data logging offered a helpful way of troubleshooting robot issues. Data stored during the run could be analyzed to determine which sensor, motor, or movement was responsible for causing problems on the run.

At first, the team only used RPS to receive the signal regarding which fuel type was necessary, but before the individual competition, the team elected to rely on RPS for navigation. Team members decided not to utilize data logging.

3.4 Testing

The OSURED team requested that each company complete four performance tests to ensure that the prototypes were making satisfactory progress. OSURED also held an individual competition on March 30 and a final competition on April 7 to examine each robot's final design and performance. The Speed D8ers conducted numerous tests on the course leading up to each performance test and competition. The testing often resulted in changes to the physical design and to the code.

3.4.1 Performance Test 1

For the first performance test, the robot was required to maneuver to the car jack, toggle the car jack to remain in the up position, and drive up a ramp to get to the upper level by February 23 [3]. Before testing, team members constructed the chassis and drivetrain to include a 6.5" by 7.5" wooden chassis, two IGWAN motors secured underneath the chassis with 3D printed mounts, two Du-Bro wheels connected to the motors with 3D printed pieces, and two additional Du-Bro wheels in the rear mounted with double bent strips and 3" axle rods, and secured in place with shaft lock collars. A picture of the robot in this stage can be found in Figure E1 in Appendix E. A CdS cell was taped to the bottom of the chassis so the robot could start when the course light was activated, as shown in Figure E2 in Appendix E.

Testing began on February 20, when team members began checking the accuracy of time-based movement. Team programmers defined values for high, medium, and low motor power, and created functions for moving forward based on specified power percentages and

time-based distances, and for turning based on specified angles. Using trial-and-error methods, the team determined distances to input into drive functions in the code in order to navigate successfully to the car jack. While testing on February 20, the team used the course strategy depicted in Figure F1 in Appendix F. The robot was programmed to move forward once the starting light was activated, turn right 90 degrees after tapping the control panel, turn left before the wrench, and then hit and align with the wall. While in the corner between the jack and the wrench, the robot made numerous 90 degree turns and aligned with both walls to straighten out before and after toggling the jack. Then, it proceeded up the grass ramp to complete the final performance test task. The robot was able to navigate using this method, but it was very inconsistent.

In an attempt to improve the consistency, team members tried using shaft encoding to navigate with the same course strategy, which is described above. After a couple tests with shaft encoding, the team remained unsatisfied with the robot's consistency. Throughout the testing period, team members noticed that the back wheels were loose, and the shaft lock collars had to be re-tightened several times. Finally, the team decided to remove the back, unpowered wheels, and test the robot on the course. The robot finally moved consistently using makeshift skids rather than wheels. The extra friction caused by the back wheels was causing the inconsistency during trials, so the Speed D8ers proceeded without the wheels. From then on, plastic spoons were used as skids, as shown up close in Figure E3 in Appendix E. The chassis and drivetrain used in Performance Test 1 is shown in Figure 5 on the following page.

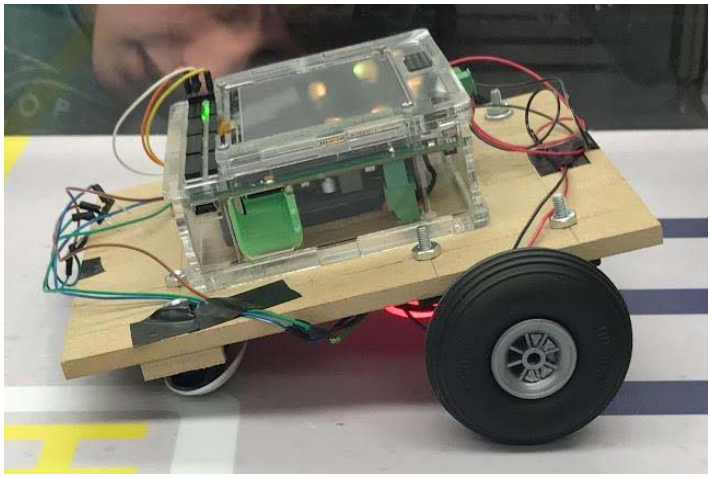


Figure 5: Robot for Performance Tests 1 and 2.

On February 21, the Speed D8ers refined the time-based program and completed additional tests with the newly-secured skids. The team conducted an official performance test using the course strategy depicted in Figure F1 in Appendix F. The robot got caught on the wall beside the ramp, and failed to complete any tasks other than touching the ramp. The team shifted its strategy thereafter to go up the other ramp instead to avoid unnecessary tight turns. This strategy is depicted in Figure F2 in Appendix F. The team used trial-and-error to come up with the correct distances to input into the sleep statements in the time-based code. Then, a second official run was conducted. The course strategy seemed to work successfully, but the jack was not toggled to remain in the up position. Team programmers quickly adjusted the distance the robot traveled to run into the jack, and a third official test was conducted. Unfortunately, the robot got caught on the wall surrounding the starting area and was not able to make it to the concrete ramp. As a result, the team re-evaluated the course strategy and decided that the tight 90 degree turns left little room for error.

On February 22, team members adjusted the program to follow the course strategy depicted in Figure F3 in Appendix F. The robot then made 45 degree turns to and from the corner near the jack to limit the number of tight turns. This code was tested several times for consistency, and then officially tested for OSURED on February 23. The performance test was successful.

3.4.2 Performance Test 2

On February 22, team programmers began working on the code for the second performance test. In this test, the robot was required to read and display the correct color of the diagnostic light, press the corresponding diagnostic button, and press the final button by March 2 [3]. The robot design remained the same from Performance Test 1, as depicted in Figure 5 on the previous page.

The course strategy for this performance test is depicted in Figure F4 in Appendix F. First, the robot turned right to touch the wrench (which proved that the team was ahead of schedule for bonus consideration from OSURED), then it moved back to align with the wall beyond the control panel. The robot then drove over the diagnostic light to read it, turned left toward the control panel, and drove into the white RPS button while turning into the correct diagnostic button. To accomplish this task, team programmers created a function to read the value detected by the CdS cell, which read less than 0.80 V if the light was blue. If the function returned a “true” value, suggesting that the diagnostic light was blue, then a separate function programmed the robot to turn into the blue button. Otherwise, if the CdS cell read greater than

0.80 V, the robot was programmed to turn into the red button. After the diagnostic button was pressed, the robot backed up and turned toward the final button to finish the performance test.

On February 23, the team tested the consistency of the code and noticed that the red button was not completely pressed. Team programmers revised the code to increase the power of the turn into the red button. Then, the team conducted an official test for the OSURED team. On the first test, the diagnostic light was blue, which the robot successfully detected and displayed on the Proteus screen. However, it did not press the button with enough power. The team conducted a second official test, where the diagnostic light was red, and it worked successfully.

3.4.3 Performance Test 3

On February 26, the team began working on the third performance test. In this test, the robot was required to pick up the wrench and deposit it in the garage by March 9 [3]. The Speed D8ers used the course strategy depicted in Figure F5 in Appendix F, moving directly from the starting position to the wrench, then directly to the garage, and finally to the fuel crank.

In order to control the wrench, the robot needed an additional mechanism. Team members constructed a pronged mechanism made out of a double angle strip and popsicle sticks, which was controlled using a Futaba servo motor. The servo motor was taped to the front of the robot on top of the wooden chassis for the test. Initially, the popsicle sticks were attached to the double angle strip with screws (as shown in Figure E4 in Appendix E), but the sticks did not stay in place, so electrical tape was found to be a better adhesive. The team also leveled the robot, which was previously angled following the removal of the back wheels, by adding four 6.5” x 1” x 0.25” pieces of wood underneath the chassis. This is depicted in Figure E5 in Appendix E.

Finally, a QR code stand was added using two 5.5” double angle strips and additional wood. A picture of the robot in this stage is shown in Figure 6 below.

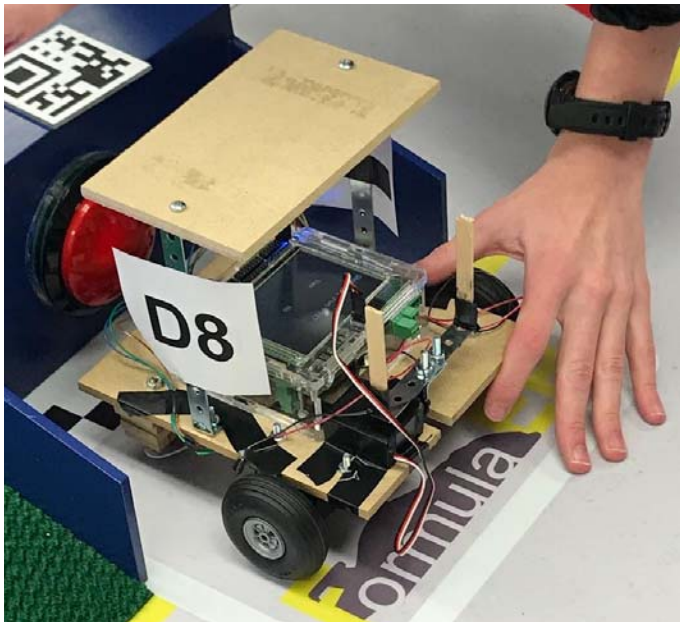


Figure 6: Robot for Performance Test 3.

Team programmers continued to rely solely on time-based movement to complete this test. Trial-and error was used to determine correct distances for the sleep statements within the code, so numerous tests were conducted on March 1 to ensure that the distances to the wrench and the garage were precise. Additional testing was performed to adjust the movement of the servo to the correct height of the wrench when grabbing it, to ensure that an appropriate power level was used to get up the ramp, and to avoid hitting the control panel.

On March 2, the team tested the consistency of the program before conducting an official test. On the first official test, the robot’s wrench mechanism did not successfully enter the holes of the wrench to grab it. The Speed D8ers conducted a successful unofficial test before running a second official test. In the second official test, the robot was too far to the right when attempting

to deposit the wrench, so the wrench was only partially deposited. Finally, on the third official test, the robot performed all necessary tasks successfully and even touched the fuel crank to earn bonus consideration from the OSURED team. Following the test, team members discovered that one possible source of error in failed runs was that the capacitors on the IGWAN motors were bent and rubbing against the motors. This possibly caused the left and right motors to run inconsistently. As a result, team members became more vigilant in checking the motors.

3.4.4 Performance Test 4

The Speed D8ers began working on the fourth performance test on March 7. For this test, the robot was required to read in the fuel type with RPS and spin the crank 180 degrees in the corresponding direction by March 23 [3]. The team added a mechanism to turn the fuel crank in order to complete the test. Team members designed a 3D printed part with a circular base and two prongs to fit inside the fuel crank, as shown in Figure E6 in Appendix E. The part was screwed into a disk that attached to a Futaba servo motor. The motor was taped to three 2.5”x1.5” pieces of wood that were screwed onto the chassis to ensure that the mechanism was at the necessary height to spin the fuel crank. A picture of the robot in this stage is shown in Figure E7 in Appendix E.

The Speed D8ers began testing the code for Performance Test 4 on March 9 using the course strategy depicted in Figure F6 in Appendix F, where the robot took the concrete ramp to the upper level and followed the concrete path before turning towards the fuel crank. The team relied on time-based movement, and conducted numerous tests on the course to determine the necessary distances, angles, and power levels needed to complete the performance test. Overall,

the team decided that using the grass ramp would lead to more consistent runs, so the course strategy depicted in Figure F7 in Appendix F was tested. During initial trials using the modified code for the new course strategy, the robot bumped into the wrench. Team members then decided to use the code for Performance Test 3 to get up the ramp with 45 degree turns rather than 90 degree turns. Team programmers modified the code from the third performance test to resemble the course strategy depicted in Figure F8 in Appendix F. The team needed to align the robot to avoid the slight dip in the course on its way to the fuel crank before conducting an official test. While testing, the team noticed that the prongs on the fuel crank mechanism were hindering its ability to turn the crank when the robot was not exactly aligned. Thus, the team removed the prongs from the mechanism and added a rubber ping pong paddle cover to the disk for increased friction. This is depicted in Figure E8 in Appendix E.

The robot did not go up the grass ramp consistently during tests, so team members altered the route to make the robot straighten up against the wall before turning towards the fuel crank. The servo also had trouble rotating the crank the full 180 degrees, so team programmers had the robot move backwards after the first rotation, reset, and spin the crank again. This strategy is depicted in Figure F9 in Appendix F. The team tested the route for consistency on March 21 before conducting official tests on March 23. In the first official test, the robot spun the crank adequately on the first rotation, but then proceeded to spin the crank further, exceeding 180 degrees. Team members removed the second spin, resulting in the course strategy shown in Figure F10 in Appendix F. The team also removed the electrical tape that was holding the servo motors in place and hot-glued them to the chassis. The robot is shown testing for Performance Test 4 in Figure 7 on the following page.

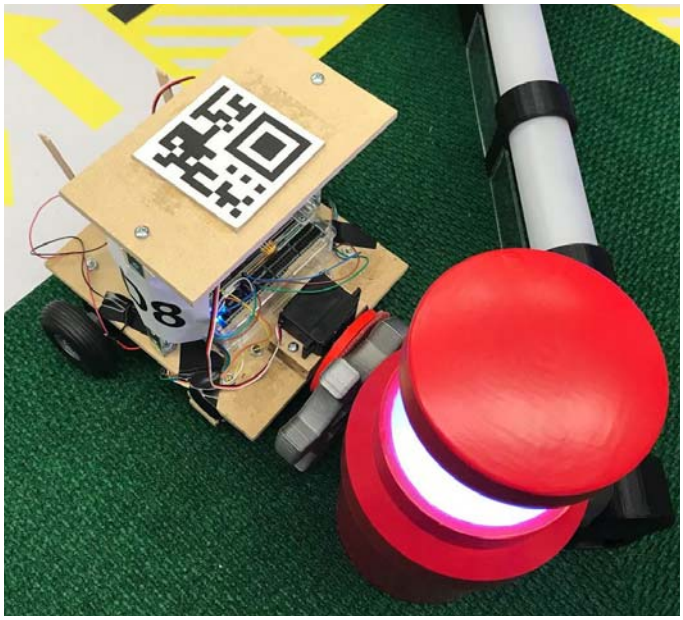


Figure 7: Robot for Performance Test 4.

The robot did not consistently spin the crank 180 degrees, but the team conducted two official tests using the revised code. On the first attempt, the crank failed to rotate the full 180 degrees, but the robot successfully completed Performance Test 4 on the second attempt.

4. Individual Competition

On March 30, 2018, the Speed D8ers' pit assistant prototype competed three full-course runs in an individual competition in Hitchcock Hall. First, the OSURED representatives selected the course, fuel type, and electrical test for the prototype. Then, the course, fuel type, and electrical test were chosen at random. For the final run, the team chose the course, fuel type, and electrical test. OSURED representatives scored each robot based on the guidelines outlined in Table 4 on the following page [2].

Table 4: Robot performance scoring guidelines.

Primary Tasks	**Points**	**Secondary Tasks**	**Points**
Initiate on start light	9	Deposit wrench	8
Touch wrench	8	Rotate fuel pump crank in correct direction	10
Control wrench	12	Rotate fuel pump crank 180 degrees	7
Toggle jack lever	10	**Possible Secondary Task Points**	**25**
Rotate fuel pump crank	8		
Press and electrical test button	7	**Total Possible Task Points**	**100**
Press the correct electrical test button	12		
Press the final charging button	9	**Penalties**	
Possible Primary Task Points	**75**	Interfering with a competitor's robot	DQ

4.1 Testing

Before the individual competition, team members hot-glued the popsicle sticks to the double angle strip and the plastic spoons to the double bent strips, labeled the various wires with tape, and added sponsor logos to the side of the chassis. The robot in this stage is shown in Figure 8 below.

Figure 8: Robot for the individual competition.

The Speed D8ers completely altered the code for the individual competition. Instead of relying solely on time-based navigation, the team utilized RPS to run the course during competition. During initial testing on March 26, the team used the course strategy depicted in Figure F11 in Appendix F, where the robot only utilized RPS on the lower level of the course and went to the control panel at the end of the run. However, team members decided that RPS would lead to greater consistency on the upper level, so they switched to the course strategy described in the next section, where the control panel was the first destination.

Tests preceding the individual competition were focused on determining x and y coordinates for the RPS checks. Team members first tested with many RPS checks to get to the desired location, leading to very slow runs. Then, team programmers added time-based code for most movements and RPS heading and location checks before turns.

4.2 Strategy

The Speed D8ers decided that the most efficient course strategy was to begin with the button control panel, pressing both the electrical test button and the RPS button while there. Then, the robot aligned with the wrench using RPS heading and coordinate checks, and moved forward to pick it up. Then, it moved back, turned left toward the wall, and turned left again before driving into the car jack. After toggling the jack, the robot backed up and turned toward the grass ramp. It made a slight turn to line up with the center of the ramp, and drove to the upper level. Then, the robot turned 45 degrees to the fuel crank. After spinning the crank, it backed up and turned 90 degrees to deposit the wrench in the garage. Then, the robot took the concrete path

back to the lower level and finished the course by pressing the final charging button. This strategy is depicted in Figure 9 below.

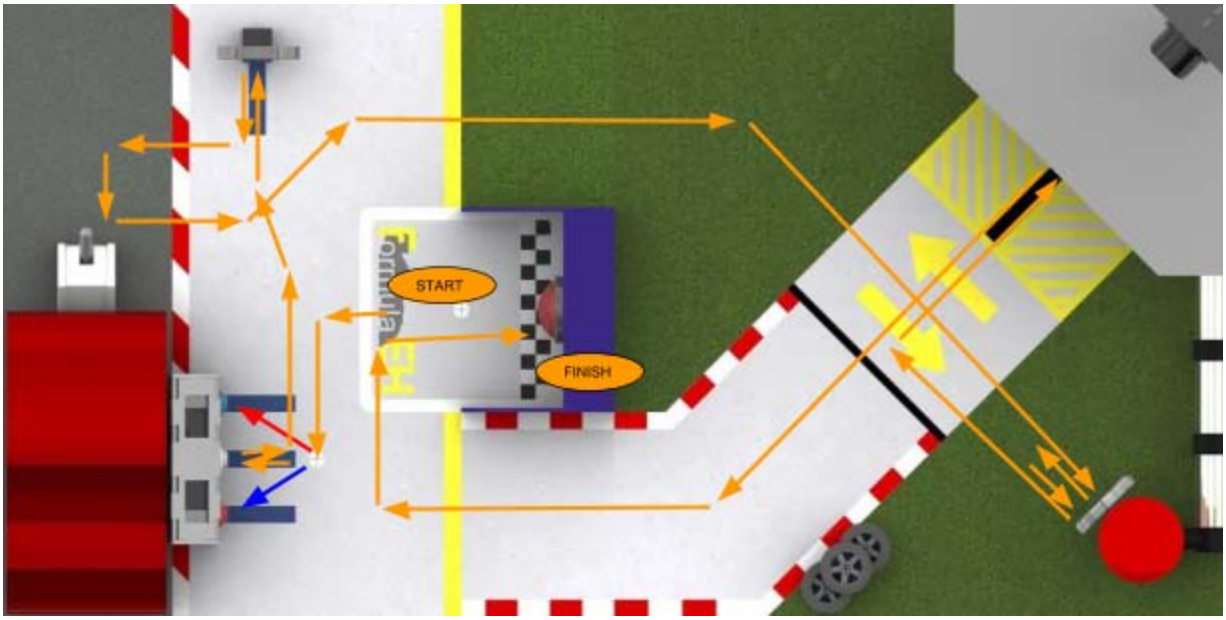


Figure 9: Course strategy for the individual competition.

4.3 First Trial

In the first run at the individual competition, OSURED representatives selected Course H, nitromethane fuel (corresponding to a counter-clockwise turn of the fuel crank), and telemetry sensors test (corresponding to the blue button). The Speed D8ers' robot successfully initiated on the start light before driving toward the diagnostic light. The robot incorrectly read the light, and it proceeded to press the red button instead of the blue button. It did press the RPS button, thus activating RPS on the upper level. The robot successfully aligned with, touched, and picked up the wrench before toggling the car jack. Then, it went up the ramp and headed toward the fuel crank. However, the prototype lined up too far to the right of the crank and turned it slightly in the incorrect direction. Then, it backed up and successfully deposited the wrench in the garage.

The robot was unable to press the final button, so the team ended the run with the kill switch after 1 minute, 19 seconds, and 68 milliseconds. Overall, the robot earned 54 points for completing primary tasks and 8 additional points for completing secondary tasks, for a total score of 62 points.

Following the run, the Speed D8ers increased three angles for the turns leading up to the final button. Team members also noticed that the prototype did not use RPS on the upper level even though it was activated. Programmers located and corrected the error in the code that caused this problem. Although the robot failed to detect the blue diagnostic light, the team did not attempt to correct this error during the individual competition.

4.4 Second Trial

In the second run at the individual competition, Course G, 98 octane fuel (corresponding to a clockwise turn of the fuel crank), and the Electrical Control Unit (ECU) test (corresponding to the red button) were randomly selected. The robot successfully initiated on the start light before driving toward the diagnostic light. It correctly read the light and pressed the red button, but not the RPS button. The robot successfully aligned with, touched, and picked up the wrench before toggling the car jack. Then, it went up the ramp and headed toward the fuel crank. However, the prototype again lined up too far to the right of the crank and turned it slightly in the incorrect direction. Then, it backed up and successfully deposited the wrench in the garage. The robot successfully pressed the final button, but it rubbed against the wall on its way down the concrete ramp. The run lasted for 1 minute, 12 seconds, and 10 milliseconds. Overall, the robot

earned all 75 points for completing primary tasks and 8 additional points for completing secondary tasks, for a total score of 83 points.

Following the run, the team added two inches to the distance traveled on the concrete path before turning to go down the ramp. This was done to avoid rubbing against the wall. The team also adjusted the distance traveled to get to the red button from the diagnostic light so the robot would be closer to the white button. Otherwise, the Speed D8ers were satisfied with the code and the robot's performance.

4.5 Third Trial

In the final run at the individual competition, the team selected Course G, 98 octane fuel (corresponding to a clockwise turn of the fuel crank), and the ECU test (corresponding to the red button). The robot successfully initiated on the start light before driving toward the diagnostic light. It correctly read the light and pressed the red and white buttons, thus activating RPS on the upper level. The robot did not align with the wrench enough to pick it up, but the wrench was touched and knocked over. Then, the robot toggled the car jack before driving up the ramp and turning toward the fuel crank. However, the prototype lined up too far to the right of the crank using RPS. It did turn the crank slightly in the correct direction. Then, it backed up and drove to the garage, but it did not have anything to deposit. The robot successfully pressed the final button, but it ran into the blue diagnostic button after coming down the ramp, thus negating its points for pressing the correct electrical button earlier in the run. This run lasted for 1 minute, 22 seconds, and 72 milliseconds. Overall, the robot earned 63 points for completing primary tasks and 10 additional points for completing secondary tasks, for a total score of 73 points.

4.6 Modifications

Following the individual competition, the Speed D8ers considered implementing several modifications. First and foremost, the robot failed to align with the fuel crank on all three trials during the competition, so the team needed to modify the code on the upper level with and without RPS. The team also needed to modify the code on the robot's route from the garage to the final button due to inconsistencies and errors made during the individual competition. Previously, the team did not use RPS on this route, but team members considered utilizing the technology to maximize precision. The robot did not consistently pick up the wrench, so the team considered modifying the course strategy to pick up the wrench first, toggle the fuel jack, and then head to the control panel. The robot also struggled to press the white button to activate RPS consistently. The Speed D8ers discussed pressing the buttons with the fuel crank mechanism to increase the area pressing the buttons and potentially increase consistency. Finally, team members noticed that the left IGWAN motor was tilted, so they planned to address that area of concern before the final competition.

Ultimately, the team did not have time to create a consistent update to the code on the upper level or from the garage to the final button. Rather than altering the course strategy to pick up the wrench first, team members decided to add an RPS check to ensure that the wrench was picked up each time. The team also increased the power at which the buttons were pressed to increase the consistency of pressing the white button to activate RPS on the upper level. Finally, the team purchased an additional IGWAN mount so the left motor did not tilt, but the left motor

weakened through testing so the team spent a large portion of time before the competition attempting to correct the motor discrepancies so that the robot could get up the ramp.

5. Final Design

Throughout the duration of the project, the Speed D8ers modified the robot's physical design and code to create a competitive prototype for the final competition. Both the design and code had strengths and weaknesses. On the way to the final design, the team monitored the budget and followed a schedule, while also logging the time spent working on each area of the project.

5.1 Features

As the project progressed, changes were continually made to the original designs, culminating in a fully functional final design. An exploded assembly of the final robot design is shown in Appendix G, along with a complete drawing set of the robot's mechanisms. The final chassis, drivetrain, electrical systems, and mechanisms are described in the following sections.

5.1.1 Chassis

The chassis of the final designed was made of MDF Engineered plywood. Its dimensions were 6.5" x 7.5". The material provided an affordable and sturdy structure, and the scrap wood was repurposed for other supporting roles on the robot. Attached to the bottom of the wooden chassis were four wooden beams supporting two wooden spoons. IGWAN motor mounts were screwed into the chassis from the underside. A CdS cell was taped to the underside of the chassis

in order to detect the lights on the course. Two long erector pieces connect the chassis to a rectangular piece of wood raised approximately 8” above the course surface. The QR code was mounted to the this upper wooden plate. The servo motor for each mechanism was glued to the front and back of the chassis. Finally, the Proteus sat on the top side of the chassis, surrounded by screws and other components so that it could not fall out. Additionally, black construction paper was attached to the wooden chassis and the QR mount overhead to give the robot a sleek appearance. Black sharpie was used to color in the edges of the chassis to create an all black robot, and the Speed D8ers emblem was glued to the center of the chassis, underneath the Proteus. A few sponsors and logos were printed on construction paper and mounted to the erector pieces supporting the QR platform, so that the logos were facing outward. The final chassis is shown in Figure E9 in Appendix E and illustrated in Appendix G.

5.1.2 Drivetrain

The final drivetrain consisted of two IGWAN motors attached to two 2.5” diameter DuBro wheels. The proposed non-powered back wheels were replaced by plastic spoons serving as skids. The spoons experienced relatively low levels of friction with the field surface. The skids were taped and glued to double bent erector pieces, where were screwed into four wooden beams lowering from the underside of the chassis. A sketch of the drivetrain is shown in Figure 10 on the following page, while drawings are displayed in Appendix G.

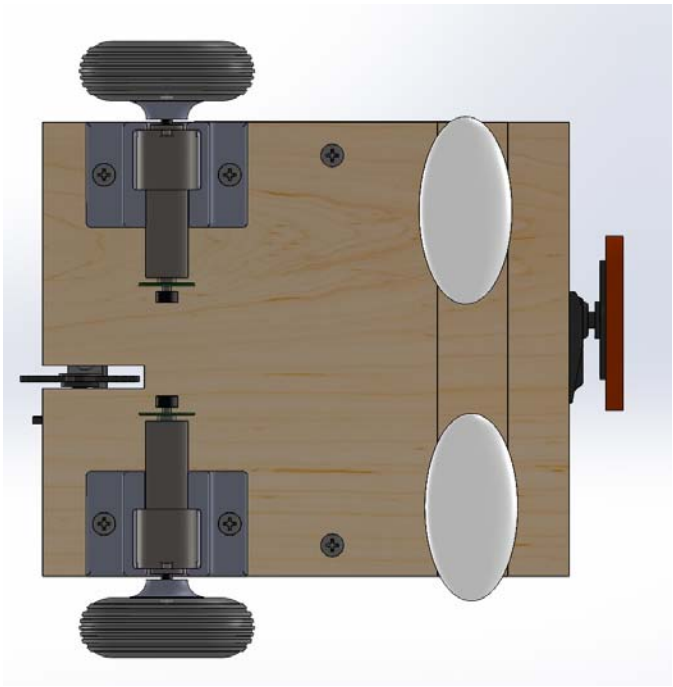


Figure 10: SolidWorks assembly of the final drivetrain.

5.1.3 Electrical System and Sensors

A total of three sensors were involved in the final design. A CdS cell was taped to the underside of the chassis. The cell was used to detect the red start light, initiating each run, and to determine the color of the floor light corresponding to the red or blue buttons. Also, both IGWAN motors contained built-in shaft encoders, but they were not used and instead taped underneath the chassis. The electrical connections for the GPIO, Servo, and Motor ports can be found in Tables H1, H2, and H3 in Appendix H, respectively. Each wire was labeled and a diagram displaying the Proteus and each connection can be seen in Figure H1 in Appendix H.

5.1.4 Mechanisms

Two mechanisms were used to complete the necessary tasks on the robot course. The first mechanism was used to transport the wrench. It consisted of a servo motor attached to a

double-bent erector piece. Two half popsicle sticks were glued to the erector piece, creating a two-pronged fork. The prongs fit through the holes of the wrench, and the servo would rotate to lift the wrench. Once the robot reached the garage, the servo would lower, causing the prongs to push the garage door open and the servo to drop the wrench. The mechanism was attached to the front of the chassis by gluing the servo to the wooden frame. A SolidWorks assembly of the mechanism is shown in Figure 11 below, and a drawing of the wrench mechanism is depicted in Appendix G.

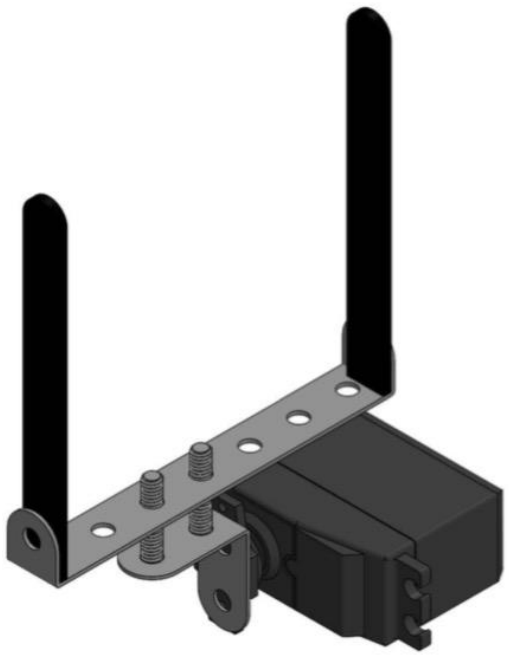


Figure 11: SolidWorks assembly of the robot's wrench mechanism.

The second mechanism was used to spin the fuel crank. It consisted of a servo motor attached to a 3D printed circular piece. A circular patch from a ping-pong paddle was adhered to the outer surface of the disk to increase the friction between the disk and the fuel crank. Before the robot reached the fuel crank, the servo would rotate to the left or right to allow it 180 degrees

of motion in the correct direction, according to the type of fuel required. Once pressed up against the fuel crank, the servo would rotate, thereby spinning the fuel crank. A picture of the mechanism can be found in Figure E8 in Appendix E, a SolidWorks assembly is shown in Figure 12 below, and a drawing of the fuel crank mechanism can be seen in Appendix G.

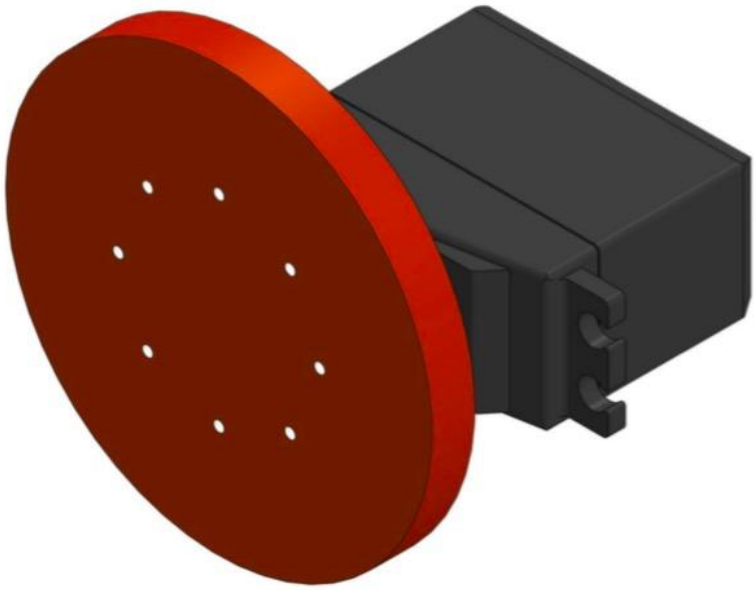


Figure 12: SolidWorks assembly of the robot's fuel crank mechanism.

Mechanisms were not needed for the car jack or any buttons. The car jack was raised by simply driving the robot into it. The buttons were pressed by driving the robot into them with the edge of the wooden chassis.

5.2 Code

The code for the robot was created using QT Creator and the Proteus, and it is provided in Appendix I. The script started with the main function, which ran a function to start the robot, then another function to run the course. The startup function took in user input, figuring out if it

should run the motor testing script, as well as which course should be run. The motor testing script ran each motor, letting the user see if any of the motors were not working properly. The function to run the course took in the course specified earlier in the startup function and ran that course. This was done through turn functions, movement functions, and functions specifically designed for completing a task.

The first function that was run was the readyStart function. This function either waited for the starting light, or, if a starting light was not detected, started in thirty seconds. After this, the fuel servo was set to the corresponding angle with another function based on the direction that needed to be turned. Two types of movement and turn functions were used. One of these used sleep statements, and the other function used RPS. The sleep statements were used for longer movements, while the RPS was used for short, quick corrections so that the robot knew where it was.

The control panel had two functions specifically for completing that task. The first function read the color of the light, assuming that the robot is above the light, and the second function used this information to move to the correct spot and hit the corresponding button. A separate function was used for the fuel crank as well, which turned the crank in a direction based on the RPS information it collected at the lower section of the course.

All other tasks, such as toggling the car jack, picking up and depositing the wrench, and pressing the final button, were accounted for within the main function to run the course through a series of straight movements and turns. Before the final competition, the team added a check using RPS to the code that corresponded to picking up the wrench. This code reviewed the RPS x-position and if it was close enough to the side wall, then the robot had successfully picked up

the wrench. Otherwise, the robot was programmed to back up, check the RPS y-position, and turn five degrees in the corresponding direction before reapproaching the wrench. This continued until the RPS x-position showed that the wrench was successfully picked up.

Overall, the code was a strength of the Speed D8ers' robot. Its modular form allowed for quick corrections during testing, and it was strong enough that the robot was competitive during the final competition on April 7, 2018. In-depth representation for each function and test is provided in Appendix J.

5.3 Budget

The team was provided with a budget of $160 to construct the robot. Plans were drawn out for initial construction, and the majority of the budget would be spent on motors and the drivetrain. Due to the decision to go with the most expensive motor option, the IGWAN, it was decided that the chassis should be constructed out of a cheaper material, MDF wood, to maintain a surplus in the budget. The first order placed cost the most money, coming in at a total of $75.96. This order cost the most because the original construction of the robot came first, and all orders placed after were for accessories or additional mechanisms. A graph of money spent over time can be seen in Appendix K as Figure K1 [4]. The breakdown of what was spent was divided into categories of motor, drivetrain, fastener, chassis, erector pieces, electrical, services, miscellaneous parts, and additional charges. How the budget was proportioned to these different categories of objects can be seen in Figure 13 on the following page [4].

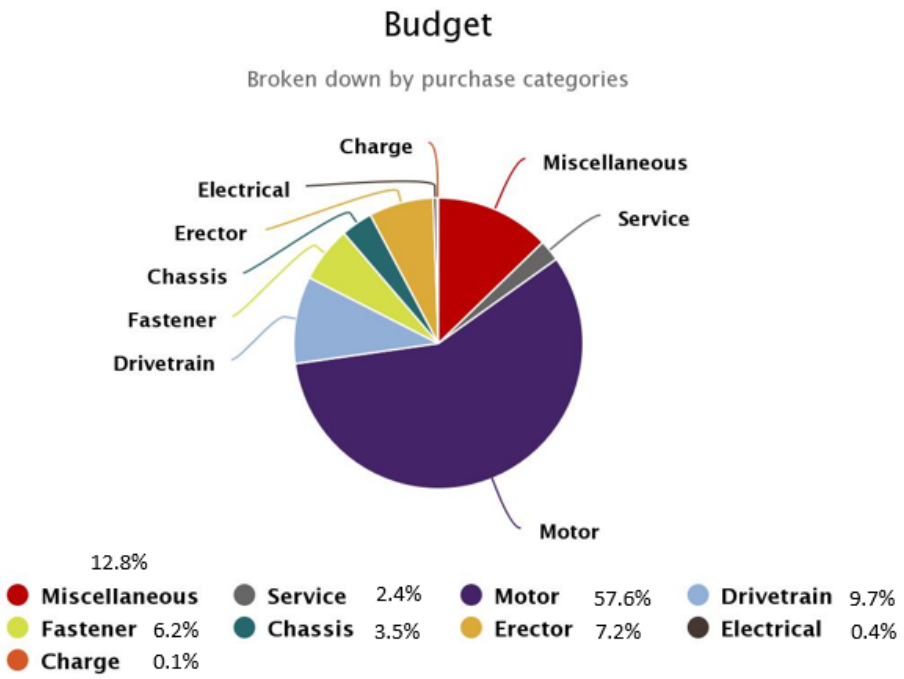


Figure 13: Breakdown of purchases by category [4].

The initial team plan was to end up with at least $40 left over, so there would be money available in case anything broke or needed to be repaired. After the team purchased all materials for the full robot, there was $44.96 remaining in the budget. Then, it was necessary to purchase two extra wheel adapters, as the spokes broke off due to wear down over time, as well as an additional IGWAN motor mount. The team finished with $38.28 remaining in the budget, meaning $122.72 was spent to construct the robot. Overall, the Speed D8ers met their goal of maintaining the budget and keeping a reserve of at least $40. A complete list of purchases and charges can be found in Table K1 in Appendix K [4].

5.4 Schedule

To ensure a timely completion of the robot prototype, the Speed D8ers produced a design schedule to plan when each task would be completed. While working on the prototype, the team kept a time log to record the time each team member spent on each part of the project.

5.4.1 Design Schedule

After forming the company, team members produced a list of primary and secondary tasks to complete, projected start and finish dates for each task, predicted the time that each task would take, and assigned primary and secondary contributors for each task. As the project was completed, team members updated the design schedule with actual start and finish dates, and the actual time each task required. The full design schedule is shown in Table L1 in Appendix L.

For the most part, the Speed D8ers stuck to the original schedule. Based on the level of difficulty of each task, some were finished early while others were not finished until the day they were due. For example, the team completed Performance Tests 1 and 4 on the due dates, a day after the projected finish date, yet they finished Performance Tests 2 and 3 each a week in advance. The team did not project the number of hours to complete each task with great accuracy, especially on larger tasks related to testing and documentation.

5.4.2 Time Log

Throughout the duration of the project, each team member maintained a time log to track how much time was spent on each portion of the project. Time was spent on project

management, documentation, coding, building, CAD modeling, testing, and other activities. Team members filled out a log each week documenting which times they worked on tasks in each category. Time logs for each week are provided in Table L2 in Appendix L. An overall time log is shown in Table 5 below.

Table 5: Overall time log.

Speed D8ers	Justin Banke	Jack McGinness	Kelly Meaden	Colby Williams	Category Totals
Documentation	16	12.5	48.5	3	80
Project Management	10	17.5	12	6.5	46
Coding	2	4.5	3	39.5	49
Testing	33	24.5	28.5	29.5	115.5
CAD	4	17	2	0	23
Building/Construction	15.5	1.5	8	2	27
Other	24.5	24.5	22.5	14.5	86
Team Totals	105	102	124.5	95	426.5

In addition to the weekly time logs, the Speed D8ers also kept a testing log for time spent using the courses provided by OSURED. The testing log provides the dates and times spent testing on each course, as well as who was testing the robot and what they were testing. Furthermore, the testing log provides a brief description of the result of each test on the course. The full testing log is shown in Appendix M.

6. Final Competition

On April, 7, 2018, the Speed D8ers' prototype competed head-to-head against 62 other prototypes at a final competition in the Recreation and Physical Activity Center (RPAC). The

competition consisted of three rounds of round robin play with four robots performing at a time. Then, there was a single-elimination tournament, where the robot that earned the most points moved on, and the other three robots were eliminated. In the event of a tie, the robot that completed the tasks in the least amount of time was declared the winner of that round. All scores were determined by the Course Automated Software (CAS) system, using the same point values as the individual competition, which are shown in Table 4 on page 30 [2].

6.1 Testing

Following a strong showing during the individual competition, the robot did not need many additional refinements before the final competition. The team added black paper and the team logo to the chassis so the robot's physical appearance was enhanced for the competition, as shown in Figure E9 in Appendix E.

The Speed D8ers began working to improve the code from the individual competition for the final competition on April 2. Team programmers added longer sections of time-based code in order to decrease the time spent checking for RPS positions. However, on April 3, the team was seeing a decline in consistency, so team members went back to the individual competition code and focused on improving consistency rather than speed. On April 5, the team added an RPS position check to ensure that the robot picked up the wrench on each run. If the robot did not reach a specific RPS x-position, then it backed up, turned slightly to the right or left, and tried to pick up the wrench again. This check greatly improved consistency. Team members attempted to add a similar check to align with the fuel crank, but this proved too difficult due to the angle of the crank.

On April 6, the left motor showed a weakness, causing the robot to struggle while moving up the ramp. The team added additional power to the left motor to help the robot get up the ramp without veering into the wall surrounding the starting area. On competition day, the team tested the lighting at the Recreational and Physical Activity Center and made adjustments to read the red and blue buttons. During a final practice run, the team noticed that the left motor still hindered the robot from making it up the ramp. Team members again increased the power of the left motor in an attempt to correct the issue. The team also added paper to the sides of the wheels to eliminate friction between the wheels and the wall. This is shown in Figure 14 below.

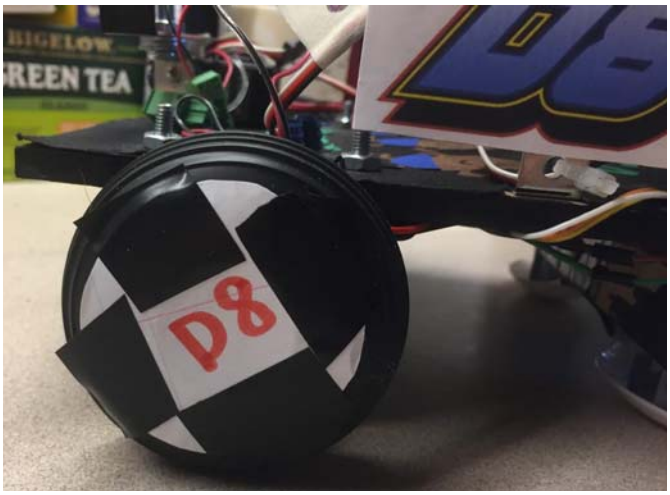


Figure 14: Paper was added to the sides of the wheels to reduce friction.

6.2 Strategy

In the final competition, the Speed D8ers planned to use the same strategy as in the individual competition, depicted in Figure 9 on page 32. The robot began with the button control panel, pressing both the electrical test button and the RPS button while there. Then, the robot aligned with the wrench using RPS heading and coordinate checks, and moved forward to pick it up. Then, it moved back, turned left toward the wall, and turned left again before driving into the

car jack. After toggling the jack, the robot backed up and turned toward the grass ramp. It made a slight turn to line up with the center of the ramp, and drove to the upper level. Then, the robot turned 45 degrees to the fuel crank. After spinning the crank, it backed up and turned 90 degrees to deposit the wrench in the garage. Then, the robot took the concrete path back to the lower level and finished the course by pressing the final charging button.

6.3 Round Robin 1

In the first round robin, the robot successfully started at the light, read the blue light, and pressed the blue button on the control panel. It also picked up the wrench, but failed to toggle the car jack. Then, the robot's left motor was too weak to make it up the ramp, causing the robot to get stuck on the starting area's wall. The team terminated the run after compiling 48 primary points and 0 secondary points in 1 minutes, 18 seconds, and 21 milliseconds.

Team programmers adjusted the motor power levels to make the robot go up the ramp with full power, but the code failed to download onto the Proteus before the second round robin. Thus, no modifications were made between the first and second rounds.

6.4 Round Robin 2

In the second round robin, the robot successfully started at the light, read the blue light, and pressed the blue button on the control panel. It also picked up the wrench, and toggled the car jack. Then, the robot again got caught on the starting area's wall before making it completely up the ramp, due to a weak left motor. The team terminated the run after compiling 58 primary points and 0 secondary points in 56 seconds and 8 milliseconds.

Following the second round, the team successfully downloaded the code to make the robot go up the ramp at full power, rather than half power with the right motor and 65-percent power with the left motor. No additional changes were made between rounds.

6.5 Round Robin 3

In the final round robin, the robot successfully started at the light, read the blue light, and pressed the blue button on the control panel. It also picked up the wrench, and toggled the car jack. Then, the robot got caught on the starting area's wall after making it completely up the ramp. This resulted in the robot being oriented parallel to the top wall of the starting area. The team terminated the run after compiling 58 primary points and 0 secondary points in 1 minute and 57 milliseconds.

Following the round robins, the team determined that the robot would be unable to make it satisfactorily to the top level without getting caught on the wall of the starting area. Thus, they had two options: not attempting to go up the ramp in order to accumulate additional points for pressing the final button immediately after toggling the car jack, or modifying the code for the upper level of the course with the assumption that the robot will get caught parallel to the starting area's wall after making it up the ramp. Ultimately, the team decided to modify the code for the upper level. Team programmers added a section of code for the robot to move straight into the side wall adjacent to the grass ramp on the upper level. The robot would then align with the wall, move forward, and then proceed with its regular program of aligning with the fuel crank using an RPS heading check. The updated strategy is shown in Figure 15 on the following page.

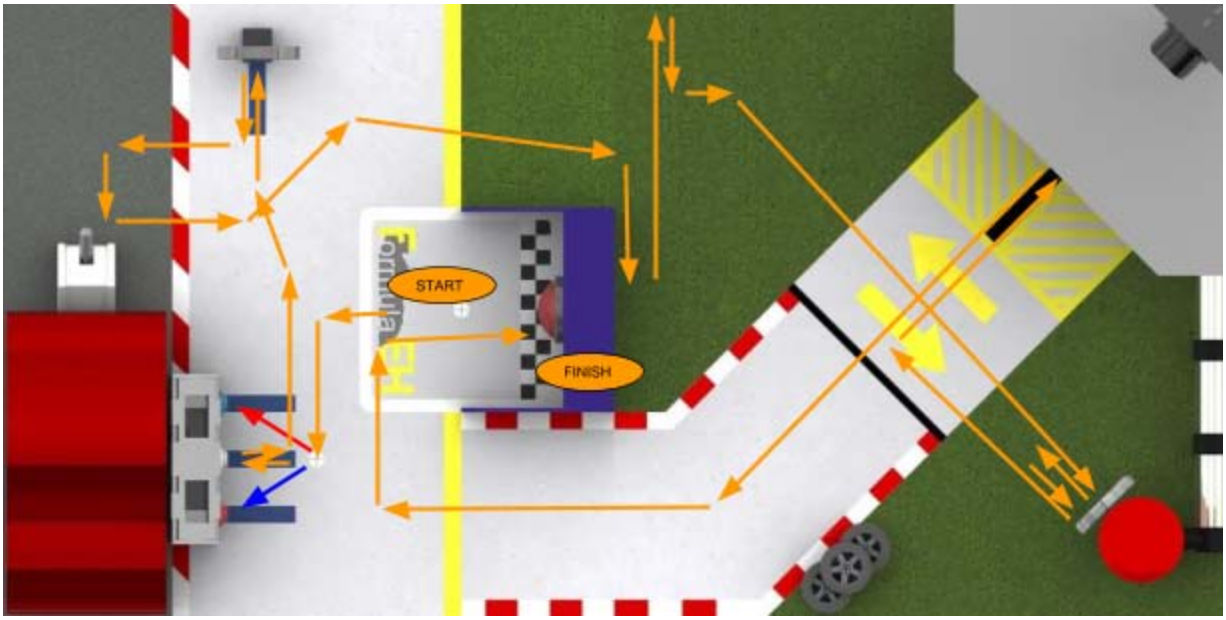


Figure 15: Course strategy for the head-to-head portion of the final competition.

6.6 Single-Elimination: Big Dance

In the first round of the single-elimination tournament, the Speed D8ers' robot successfully started at the light, read the red light, and pressed the red button on the control panel. It held the white RPS button for 5 seconds, picked up the wrench, and toggled the car jack. Then, the robot made it completely up the ramp for the first time all day. Since the team altered the code with the assumption that the robot would get caught against the wall of the starting area, the robot then turned to orient itself toward the garage instead of the fuel crank. Slowly, the robot turned to face the fuel crank using RPS heading check. The robot approached the fuel crank, aligning to the right of the center of the crank. This caused the crank to turn in the wrong direction. The, the robot moved to deposit the wrench in the garage, but the movement was too fast and the wrench was not deposited. The robot then quickly went back to the lower level to press the final button with three seconds remaining. The robot compiled all 75 primary points

and 0 secondary points in 1 minute, 57 seconds, and 31 milliseconds. Fortunately, the other three robots compiled fewer points, so the Speed D8ers moved on to the Sweet Sixteen.

6.7 Single-Elimination: Sweet Sixteen

In the Sweet Sixteen, the Speed D8ers' robot started with the light, recognized the blue light, pressed the blue button, and held the white RPS button for the full 5 seconds. It picked up the wrench, toggled the car jack, and traveled up the grass ramp. While it got caught on the starting area wall again, the robot successfully backed up against the pipe on the top portion of the course. The robot then turned off of the wall and used RPS to orient itself with the fuel crank. The robot was able to turn the fuel crank only a little less than 180 degrees in the correct direction. The robot then successfully deposited the wrench but got caught on the starting area wall before pressing the final button. In total, the robot earned 66 of the 75 primary points and 18 of the secondary points in 1 minutes, 38 seconds, and 77 milliseconds. Unfortunately, a competing team scored more than the Speed D8ers' 84 points, eliminating the Speed D8ers from the tournament.

6.8 Performance Analysis

Overall, the Speed D8ers' robot met and exceeded expectations by advancing to the Sweet Sixteen and placing 2nd in its second round match. But initially, the Round Robin matches were not satisfactory, as the robot was unable to successfully climb the ramp. In all three Round Robin runs, the robot completed the components on the lower level correctly. The only exception was the robot missing the car jack on the first round. However, imbalance in motor strengths –

which had been witnessed in the two days prior to the tournament – caused the robot to get caught on the starting area wall, prompting the team to terminate the runs. No adjustments were made between the first two runs. For the third Round Robin match, the both wheels were programmed to drive up the ramp at 100 percent speed. While the robot did make it fully to the top level, it still got caught on the starting area.

The team attempted to solve this problem by first setting the motor speeds back to the way they were in rounds 1 and 2. Assuming the robot would get stuck, it was programmed to back into the pipes along the border of the top region of the field. After squaring, the robot would move forward, turn, and use RPS checks to align with the fuel crank. While this was never tested, it proved to be successful, as the robot turned the crank, reached the garage, and pressed the final button in the first elimination match. Because of the issues faced in the first three rounds, advancing to the next round and completing all primary tasks exceeded all expectations. Furthermore, the robot performed even better on the next round, turning the crank the correct direction, albeit about 10 degrees too little, and successfully depositing the wrench, although it missed the final button as well. A score of 84 and placing 2nd in the Sweet Sixteen match was very acceptable, especially provided the difficulties face in the Round Robin.

7. Summary and Conclusions

During this project, the Speed D8ers designed, built, and programmed a robot to complete tasks in the Formula EH Grand Prix pit area. Beginning on January 26, when the project was presented by OSURED, each team member brainstormed multiple ideas to accomplish for each task. The Speed D8ers company formed on January 31, and came up with

three preliminary concepts and a mock-up. Throughout the project, the team made critical decisions through testing, analysis, and calculations while constructing and coding the robot. The robot competed in an individual competition on March 30, and the team made additional modifications leading up to the final design. Finally, the robot competed in a final competition on April 7, where it overcame motor struggles to advance to the Sweet Sixteen.

7.1 Summary

After much deliberation and brainstorming, the Speed D8ers constructed their chassis and drivetrain using MDF ¼" wood, two wheels powered by IGWAN motors and two additional unpowered wheels. The team added a mechanism to pick up the wrench using erector set pieces and popsicle sticks, powered by a Futaba servo motor. A mechanism to spin the fuel crank was made using a 3D printed disk with prongs, powered by a Futaba servo motor. The team also utilized a CdS cell to detect the color of lights on the course and a QR code to receive transmissions from the Robot Positioning System.

After further analysis and testing, the Speed D8ers removed the unpowered wheels and replaced them with plastic spoon skids. The team also removed the prongs on the fuel crank mechanism and added a ping pong paddle cover to increase the friction of the mechanism. Overall, the team spent $122.72 to construct the robot.

Team programmers initially relied on time-based navigation for the first four performance tests, but added in RPS checks for optimal performance during the individual and final competitions. During the individual competition, the robot consistently activated with the starting light, toggled the car jack, and deposited the wrench. The robot inconsistently pressed

the correct diagnostic button, picked up the wrench, and pressed the final button. The robot failed to spin the fuel crank 180 degrees in the correct direction. Before the final competition, the team added code using RPS to ensure that the wrench was picked up on every run. The left motor weakened, leading to an inability to make it up the ramp in the first portion of the final competition. The team added additional code before the head-to-head portion of the competition in an attempt to compensate for the motor struggles, and this allowed the robot to complete tasks on the upper level. Ultimately, the Speed D8ers advanced to the Sweet Sixteen before being eliminated.

7.2 Conclusions

Overall, the Speed D8ers' robot performed satisfactorily. The height of the chassis was optimal for pressing the buttons on the control panel and toggling the jack without having to add and additional mechanism to complete those tasks. Unfortunately, the IGWAN motors weakened due to excessive use, which caused some problems in the final competition. Additionally, the mechanisms for the wrench and fuel crank were inconsistent. Although the popsicle sticks were able to pick up the wrench most of the time, the robot occasionally failed to align perfectly with the wrench. The team was able to add an RPS position check to ensure that the wrench was picked up during each round of the competition, but the check hindered the overall speed. Likewise, the spinning disk had to be perfectly aligned with the crank in order to spin it 180 degrees in the correct direction, and it was often misaligned. Since the servo motor could not rotate completely, the robot had to approach the fuel crank twice to spin it the full amount.

In the future, companies should hack the servo motors so they are able to rotate 360 degrees. This would enhance the speed of the robot on the course and lead to more accuracy while spinning the fuel crank. Companies should also consider using a magnet to pick up the wrench instead of a pronged mechanism that only works when it is perfectly aligned. Additional suggestions for future changes before production of the robot include utilizing bump switches to increase the overall speed of each run, and ensuring that all motors are fully functioning. With these changes, the prototype would have a better chance of being selected as the pit assistant for the Formula EH Grand Prix.

8. References

[1] "With new pit-stop rules in place, NASCAR teams in development mode." 2018, January 24. www.nascar.com.

[2] Robot Course Scenario. 2018, January 26. www.carmen.osu.edu.

[3] Performance Tests Robot 2018. 2018, January 23. www.carmen.osu.edu.

[4] FEH Robot Store. 2018, March 31. www.feh.osu.edu.

APPENDIX A

Brainstorming

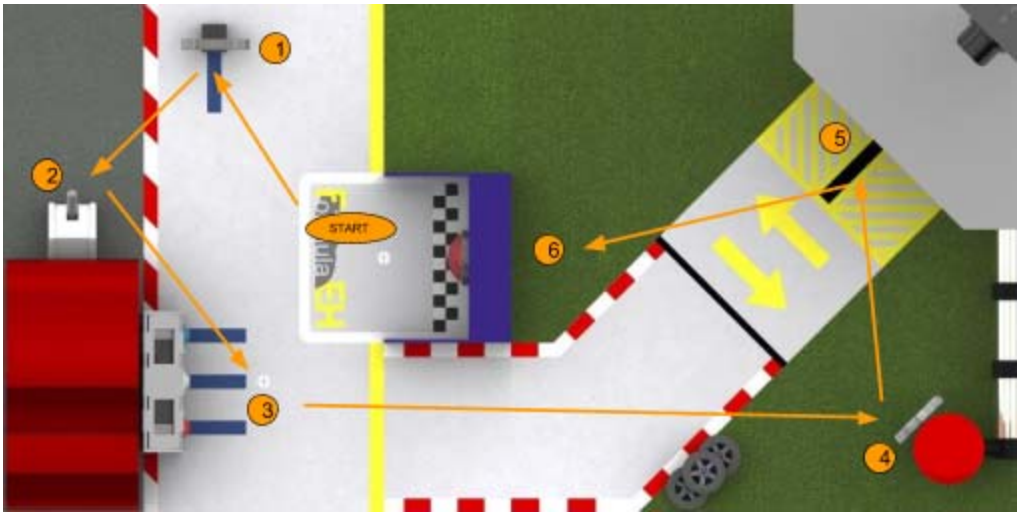


Figure A1: Second-ranked course strategy.

Table A1: Screening matrix for chassis and drivetrain brainstorming ideas.

Chassis and Drivetrain						
Success Criteria	**Ref**	**Design A**	**Design B**	**Design C**	**Design D**	**Design E**
Description		aluminum, 4 motored wheels	PVC, 2 motored wheels	Acrylic, 2 motored wheels + chain	Wood, Circular wheel layout (4)	2 robots
Speed	0	+	0	0	0	+
Power	0	+	0	0	0	-
Maneuverability	0	0	-	-	+	+
Cost	0	-	+	+	-	-
Maintenance	0	-	+	-	+	-
Durability	0	0	0	+	0	-
Sum +	0	2	2	2	2	2
Sum 0	6	2	3	2	3	0
Sum -	0	2	1	2	1	4
Net Score	0	0	1	0	1	-2
Continue?		Yes	Yes	Yes	No	No

Table A2: Screening matrix for car jack mechanism brainstorming ideas.

Car Jack					
Success Criteria	**Ref**	**Design A**	**Design B**	**Design C**	**Design D**
Description		Sloped Front	Lever	Arm With Clamp	Rod on Gear
Accuracy	0	-	+	+	0
Ease of Control	0	0	+	-	0
Speed	0	+	0	-	0
Consistency	0	0	-	+	0
Cost	0	+	-	-	-
Sum +	0	2	2	2	0
Sum 0	5	2	1	0	4
Sum -	0	1	2	3	1
Net Score	0	1	0	-1	-1
Continue?		Yes	Yes	No	No

Table A3: Screening matrix for fuel crank mechanism brainstorming ideas.

Fuel Crank					
Success Criteria	**Ref**	**Design A**	**Design B**	**Design C**	**Design D**
Description		Rod on Gear	Rotate Disk	Vertical spoke arm	Two rods on gear
Accuracy	0	0	-	+	+
Ease of Control	0	+	0	0	0
Speed	0	0	+	-	0
Consistency	0	0	0	-	+
Cost	0	0	0	0	0
Sum +	0	1	1	1	2
Sum 0	5	4	3	2	3
Sum -	0	0	1	2	0
Net Score	0	1	0	-1	2
Continue?		Yes	Yes	No	Yes

Table A4: Screening matrix for wrench mechanism brainstorming ideas.

Wrench					
Success Criteria	**Ref**	**Design A**	**Design B**	**Design C**	**Design D**
Description		Arm Go In Hole	Fork Lift	Rod with Stopper	Arm with Clamp
Accuracy	0	0	+	-	+
Ease of Control	0	+	0	+	0
Speed	0	0	-	+	-
Consistency	0	0	+	-	+
Cost	0	+	0	+	-
Sum +	0	2	2	3	2
Sum 0	5	3	2	0	1
Sum -	0	0	1	2	2
Net Score	0	2	1	1	0
Continue?		Yes	Yes	Yes	No

Table A5: Concept scoring matrix for three preliminary ideas.

		Reference		**Idea One**		**Idea Two**		**Idea Three**	
Success Criteria	Weight	Rate	Weight	Rate	Weight	Rate	Weight	Rate	Weight
Speed	20%	3	0.6	2	0.4	2	0.4	4	0.8
Power	5%	3	0.15	2	0.1	3	0.15	5	0.25
Maneuverability	25%	3	0.75	4	1	5	1.25	5	1.25
Cost	20%	3	0.6	5	1	4	0.8	2	0.4
Consistency	20%	3	0.6	4	0.8	3	0.6	3	0.6
Durability	10%	3	0.3	5	0.5	4	0.4	4	0.4
Total Score	100%		3		3.8		3.6		3.7
Continue		No		Yes		Yes		Yes	

APPENDIX B

Preliminary Ideas

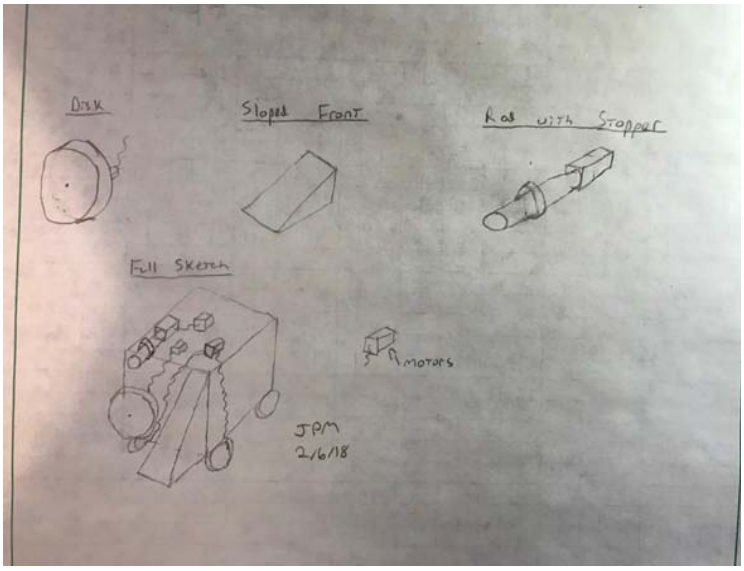


Figure B1: Two dimensional sketches depicting Preliminary Idea One.

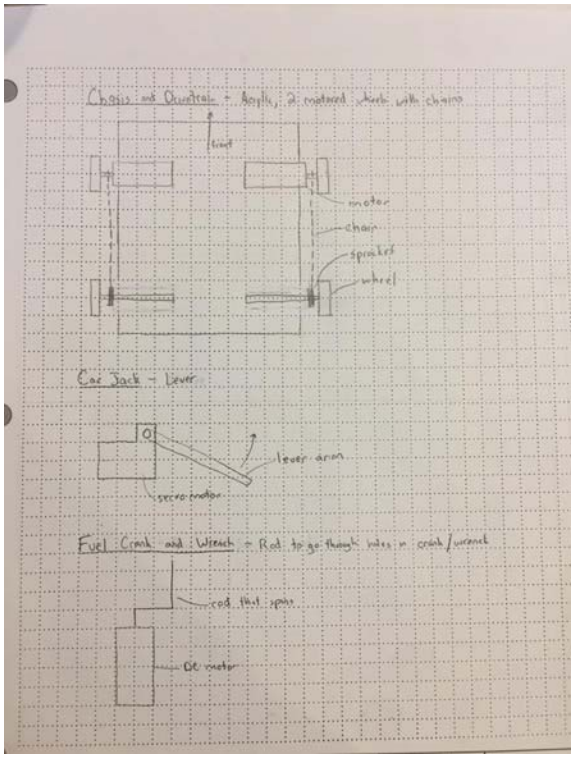


Figure B2: Two dimensional profile sketches of Preliminary Idea Two components.

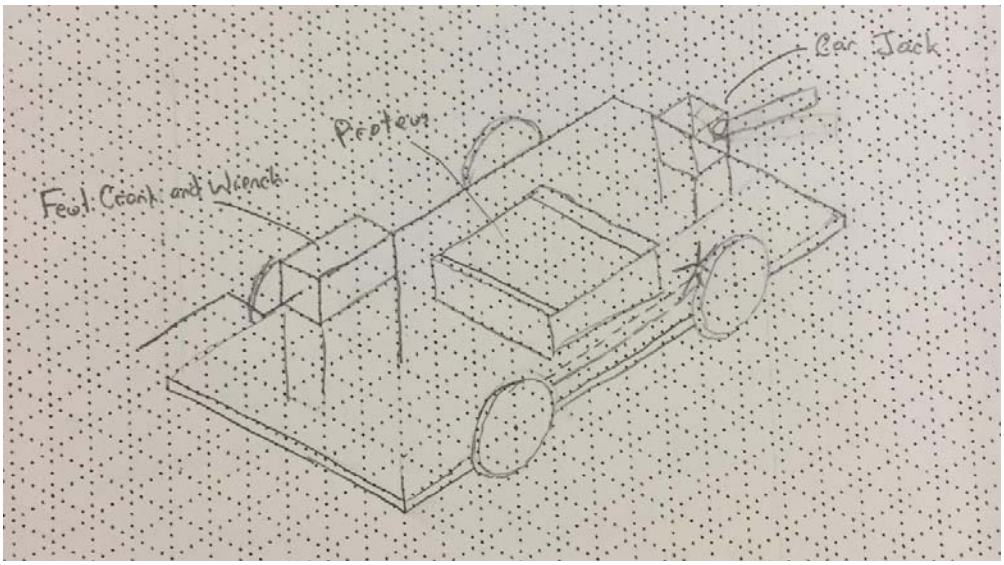


Figure B3: Isometric sketch of Preliminary Idea Two.

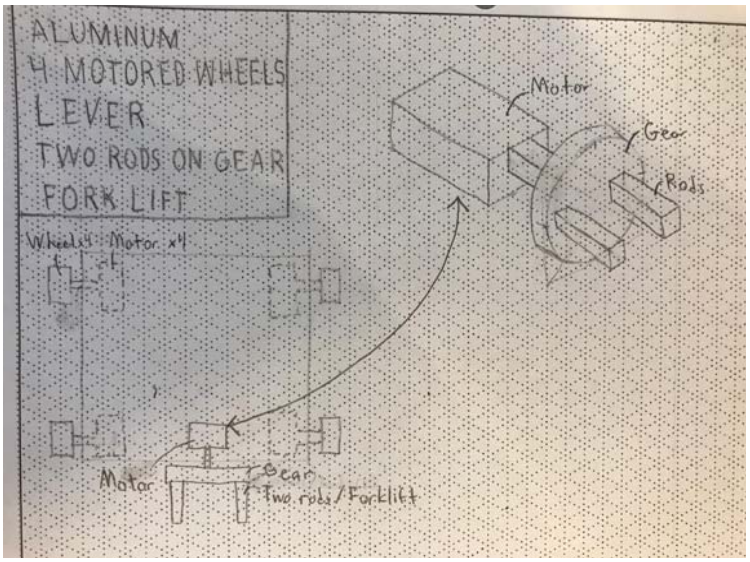


Figure B4: Sketch depicting Preliminary Idea Three.

Preliminary Algorithm:

1. Wait for the starting light to turn on
2. Drive forward until lined up with wrench
3. Turn right 90 degrees
4. Lower wrench servo
5. Move forward, grabbing wrench with servo
6. Raise wrench servo
7. Move backwards, lining up with jack area
8. Turn left 90 degrees
9. Move forwards, lining up with jack
10. Turn left 90 degrees
11. Move forwards, pushing jack up
12. Move backwards, lining up with ramp
13. Turn right 90 degrees
14. Move backwards until fully up ramp
15. Turn right until back of robot is lined up with fuel crank
16. Move backwards until flush with fuel crank
17. Rotate fuel servo based on the direction desired by the specified run
18. Move forward, lining up with garage
19. Turn right, lining up with garage
20. Move forward until flush with garage
21. Lower wrench servo
22. Move backwards until at the smooth ramp
23. Raise wrench servo
24. Turn right, lining up with ramp
25. Move backwards down the ramp until over button board light
26. Read the color of the light.
27. If the color was red, move at a right angle and push into red button
28. If the color was blue, move at a left angle and push into left button
29. Reset on light
30. Turn left 90 degrees
31. Move forward, lining up with final button
32. Turn right 90 degrees
33. Move forward until the final button is pressed

APPENDIX C

Analysis & Explorations

Table C1: Measurements of segmented navigation through robot course.

Start Location	**End Location**	**Distance (inches)**
Start Zone	Buttons	16
Buttons	Car Jack	24
Car Jack	Wrench	16
Wrench	Garage	52
Garage	Fuel Crank	25
Fuel Crank	End Button	60
	Total	**193**

Table C2: Estimated times for robot course tasks.

Action	**Description**	**Estimated Duration (seconds)**
press correct button	read color, turn right or left, drive into correct colored button and white button for 5 seconds	8
lift car jack	drive into car jack	2
pick up wrench	drive so arms go through holes, lift servo	7
deposit wrench	drive into garage, lower servo, back up	3
turn fuel crank	turn servo correct direction, drive so arms go through holes, turn servo correct direction	5
press end button	drive into end button	1
	Total	**26**

Table C3: Estimated weights for robot components.

Component	**Approximate Unit Weight (g)**	**Quantity**	**Approximate Total Weight (g)**
aluminum chassis	200	1	200
Igwan motor	71	2	142
2” DuBro wheel	20	4	80
Futaba servo motor	37	2	74
Proteus	437	1	437
CdS cell and case	15	1	15
aluminum button plate	38	1	38
PVC mechanism	6	1	6
wooden mechanism support	52	1	52
wires, screws, buffer	15	1	15
		Total	**1059**

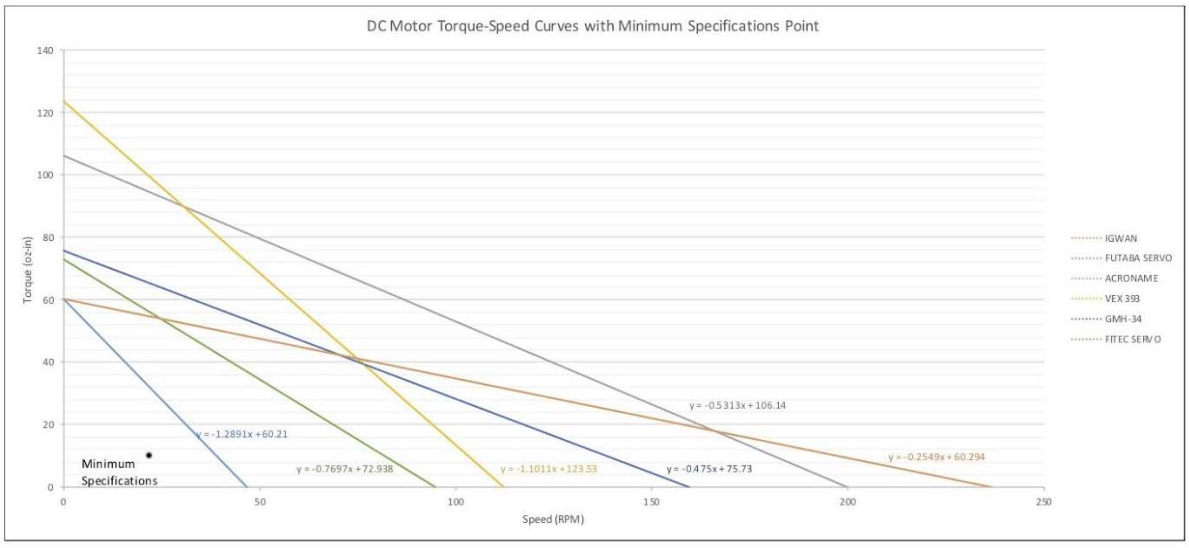


Figure C1: DC motor torque-speed curves with minimum specifications point.

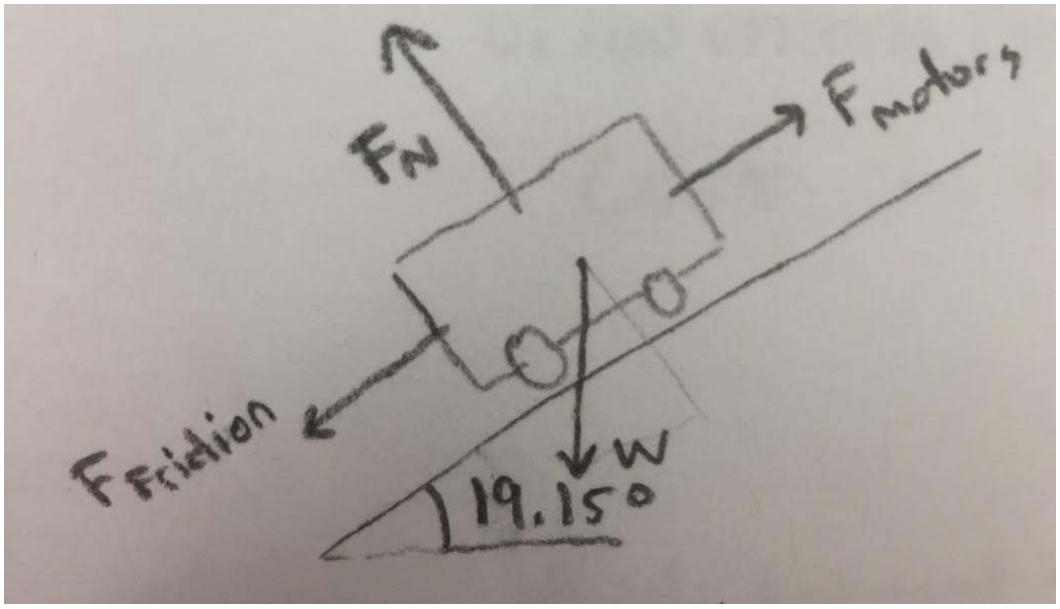


Figure C2: Free-body diagram of the forces acting on the robot traveling up the course ramp.

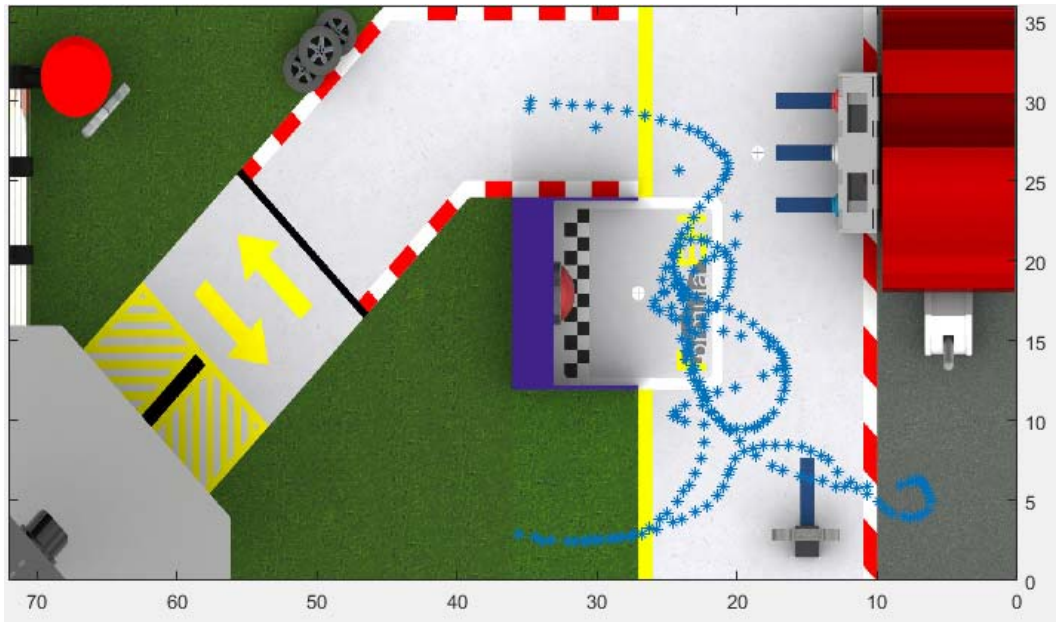


Figure C3: Data log using RPS coordinates on MATLAB.

APPENDIX D

Sample Calculations

Sample Calculation for Equation 1:

$$v = \frac{distance}{time} = \frac{193\ inches}{85\ seconds} = \textbf{2.3 in/sec}$$

Sample Calculation for Equation 2:

$$\omega = \frac{v}{2\Pi r} = \frac{2.3\ in/sec}{2\Pi \cdot 1\ in} = \frac{0.366\ rev}{1\ sec} \cdot \frac{60\ sec}{1\ min} = \textbf{22.0 rev/min}$$

Sample Calculation for Equation 3:

$$ounces = grams \cdot \frac{1\ oz.}{28.35\ g} = 1059\ g \cdot \frac{1\ oz.}{28.35\ g} = \textbf{37.36 oz.}$$

Sample Calculation for Equation 4:

$$F_{motors} = F_{friction} + W sin(19.15) = 8\ oz + (37.36\ oz \cdot sin(19.15)) = \textbf{20.26 oz.}$$

Sample Calculation for Equation 5:

$$\tau_{motors} = F_{motors} \times r_{wheel} = 20.26\ oz \times 1.00\ in = \textbf{20.26 oz-in}$$

Sample Calculation for Equation 6:

$$\frac{encoder\ units}{1\ inch} = \frac{encoder\ units}{rotations} \cdot \frac{rotations}{\Pi d\ inches} = \frac{318\ encoder\ units}{1\ rotation} \cdot \frac{1\ rotation}{\Pi \cdot 2.5\ inches} = \textbf{40.489 units/inch}$$

APPENDIX E

Design Progression

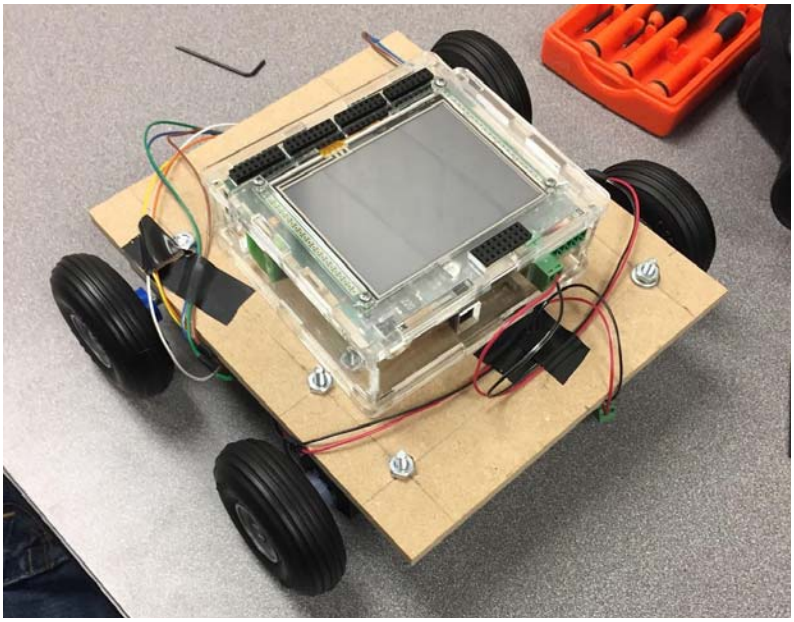


Figure E1: First build of the chassis and drivetrain.

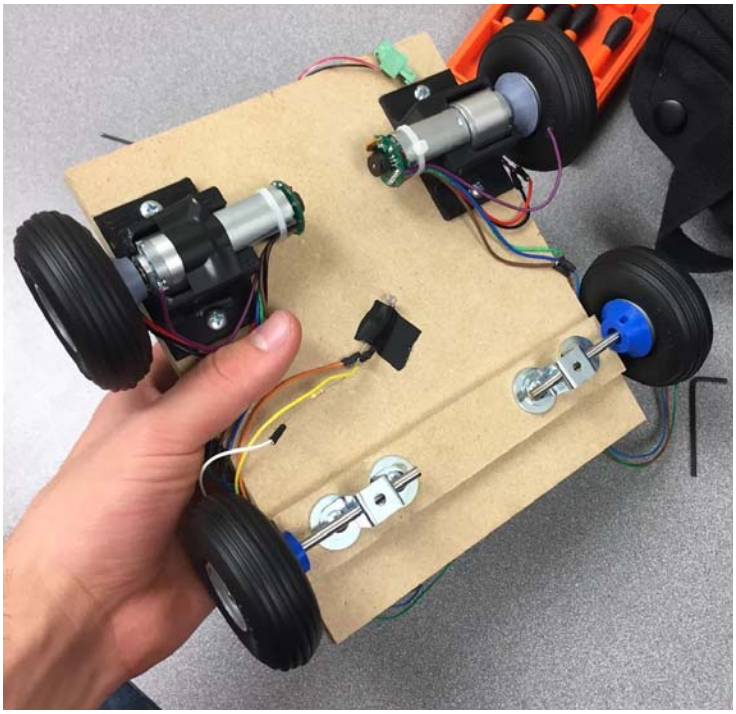


Figure E2: Bottom view of the initial build.

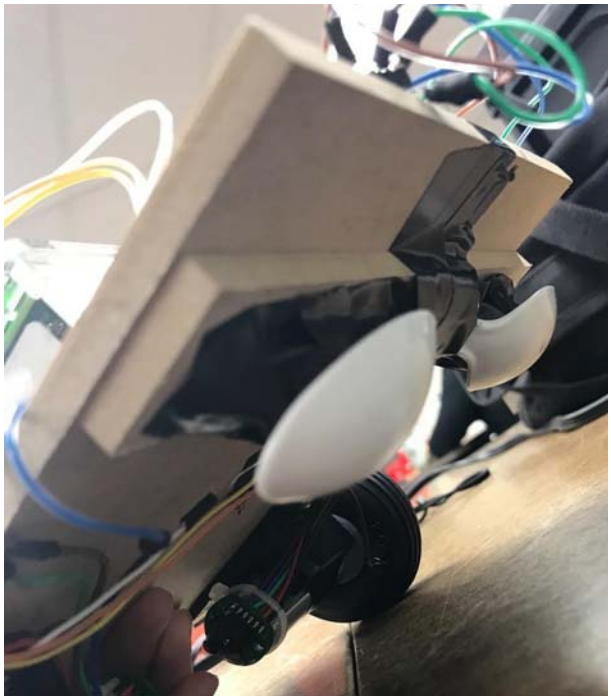


Figure E3: Plastic spoons added as skids, replacing the back wheels.

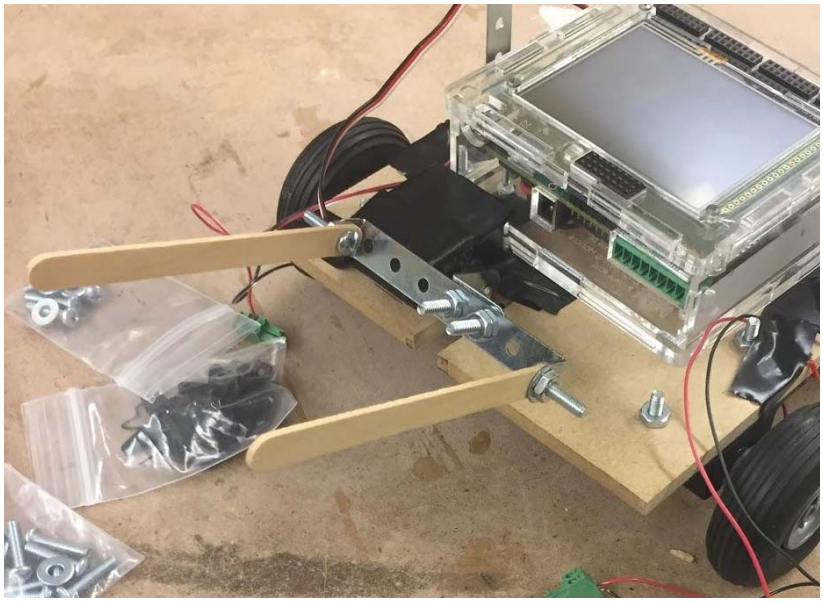


Figure E4: Initial construction of wrench mechanism.

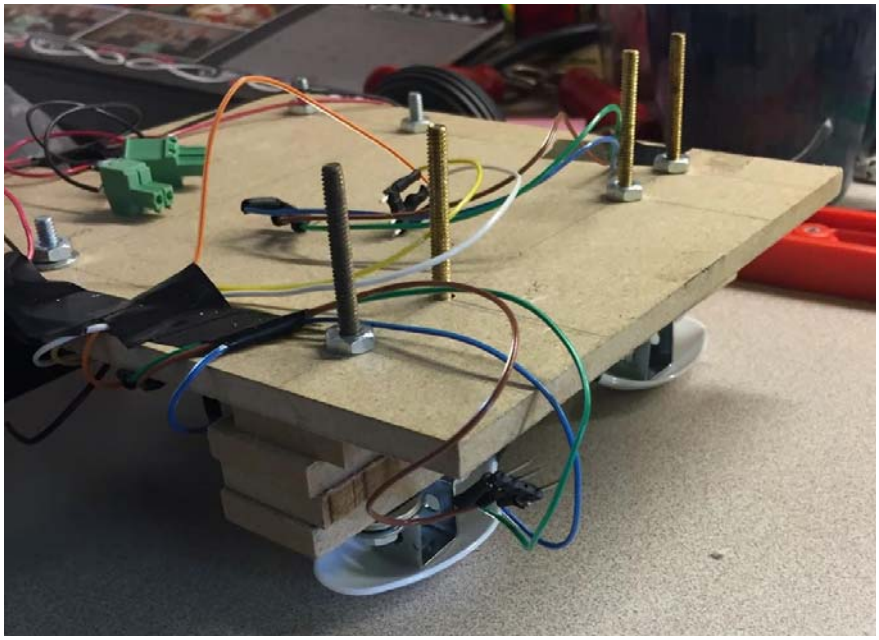


Figure E5: Four pieces of wood underneath chassis to level the robot.

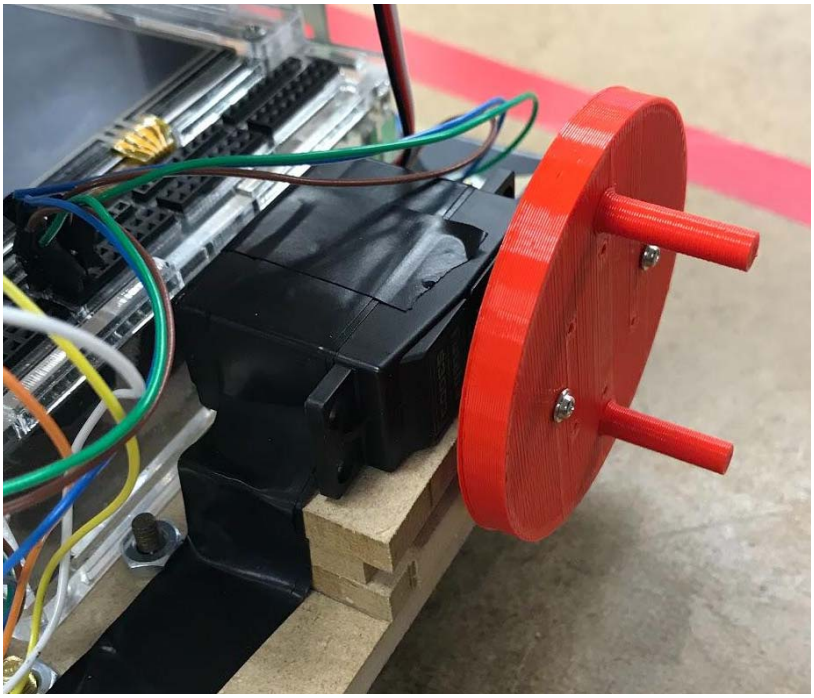


Figure E6: 3D printed part for the fuel crank mechanism.

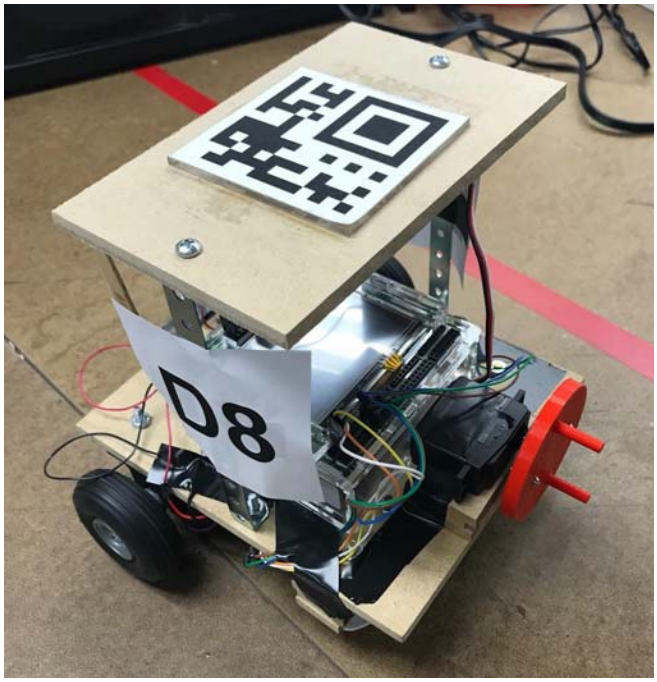


Figure E7: Robot with pronged 3D printed fuel crank mechanism.

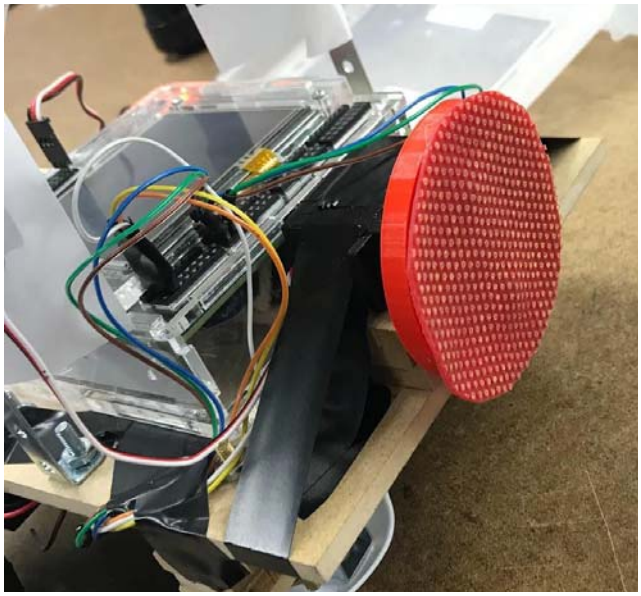


Figure E8: Revised fuel crank mechanism for Performance Test 4.

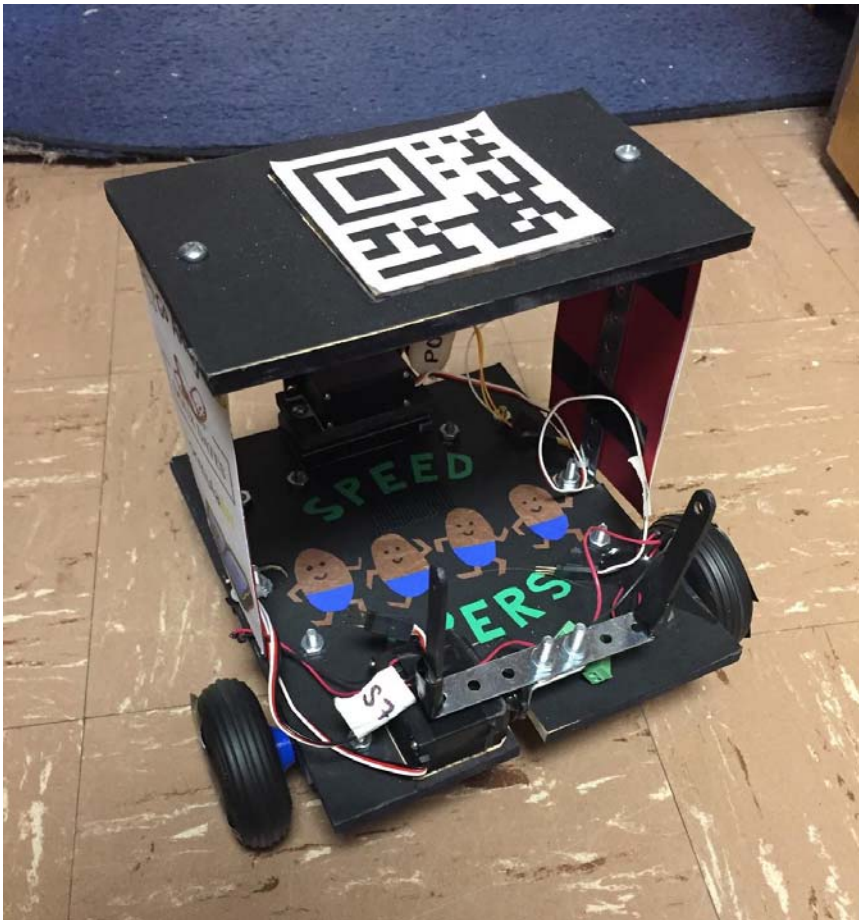


Figure E9: Robot for final competition.

APPENDIX F

Testing Strategies

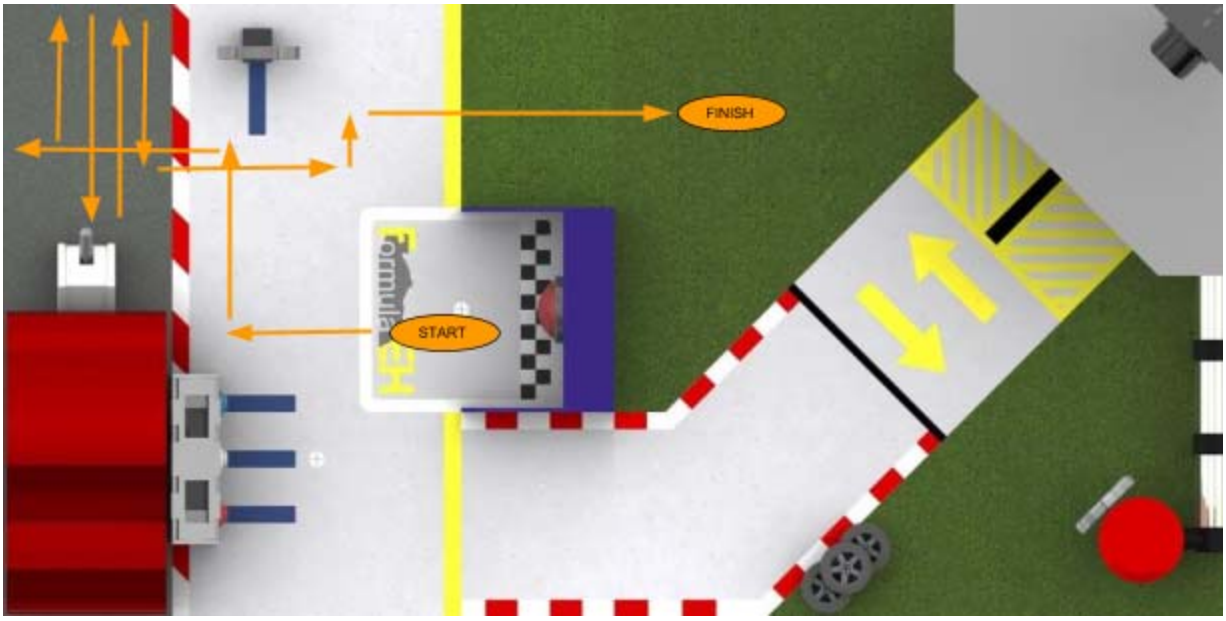


Figure F1: Initial course strategy for Performance Test 1.

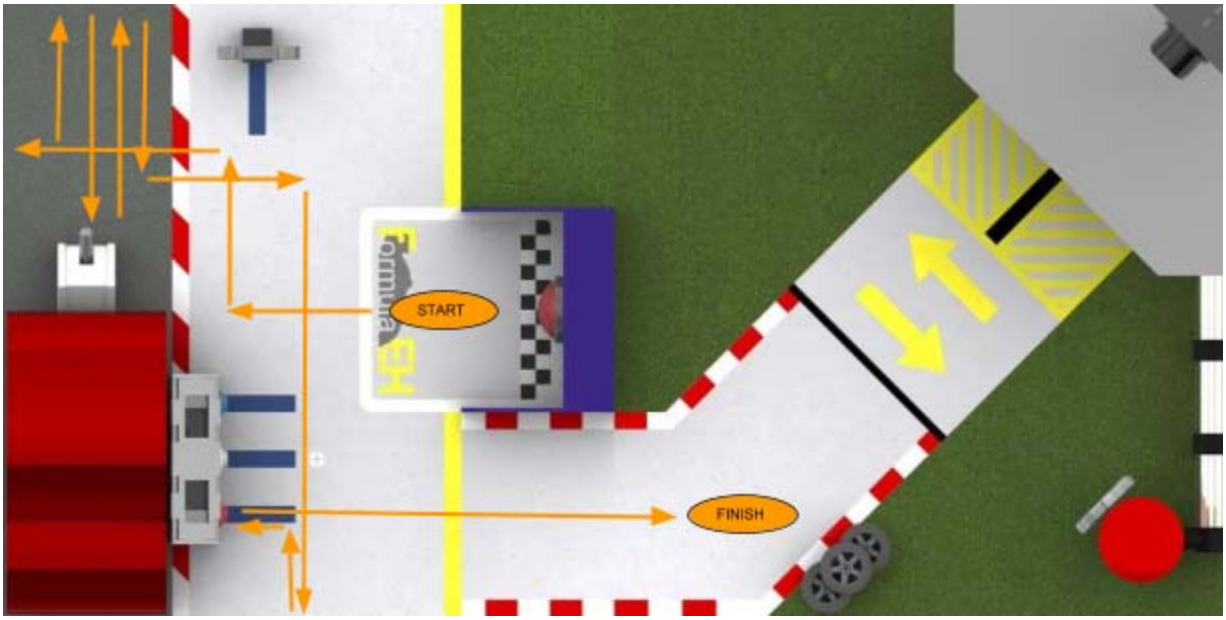


Figure F2: Second course strategy for Performance Test 1.

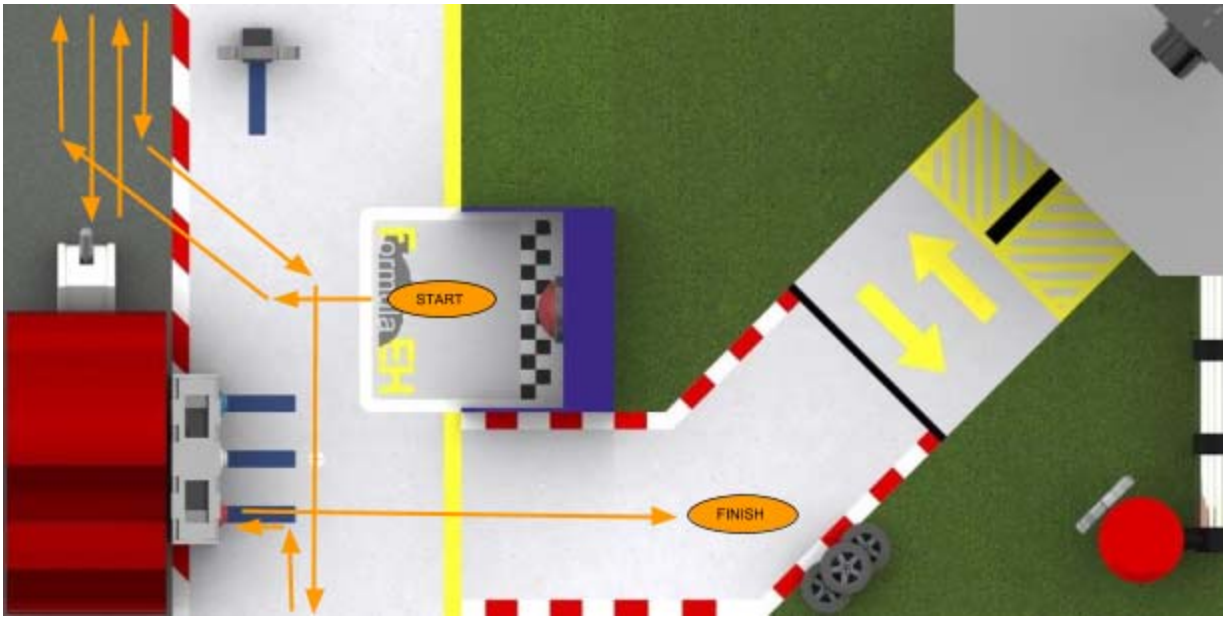


Figure F3: Final course strategy for Performance Test 1.

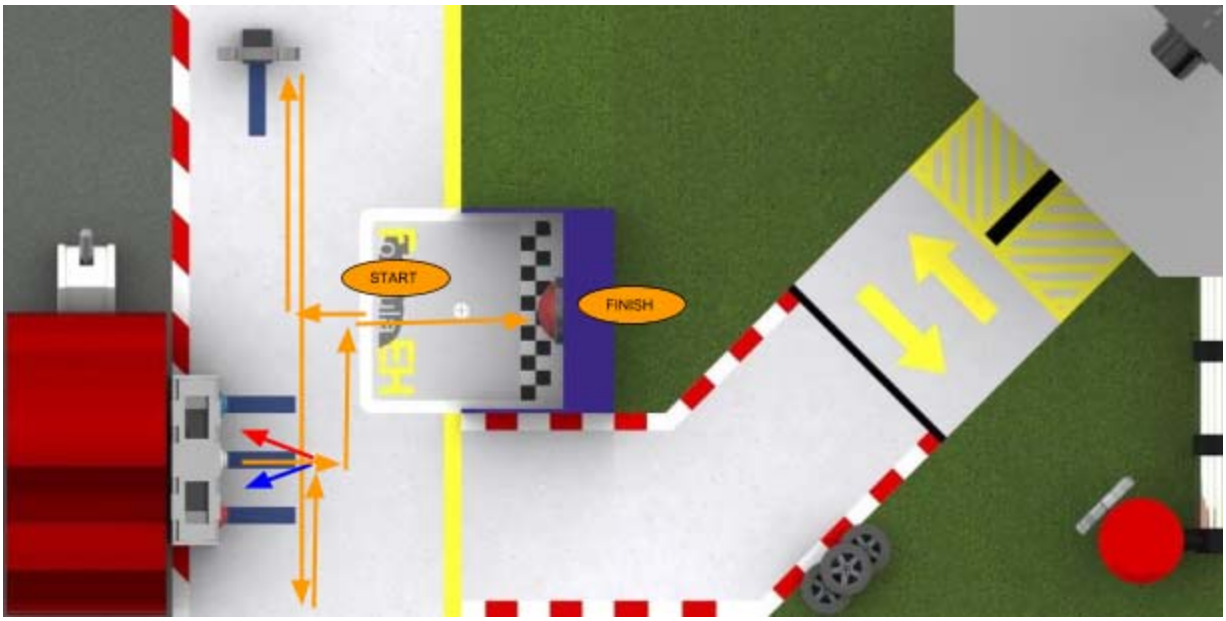


Figure F4: Course strategy for Performance Test 2.

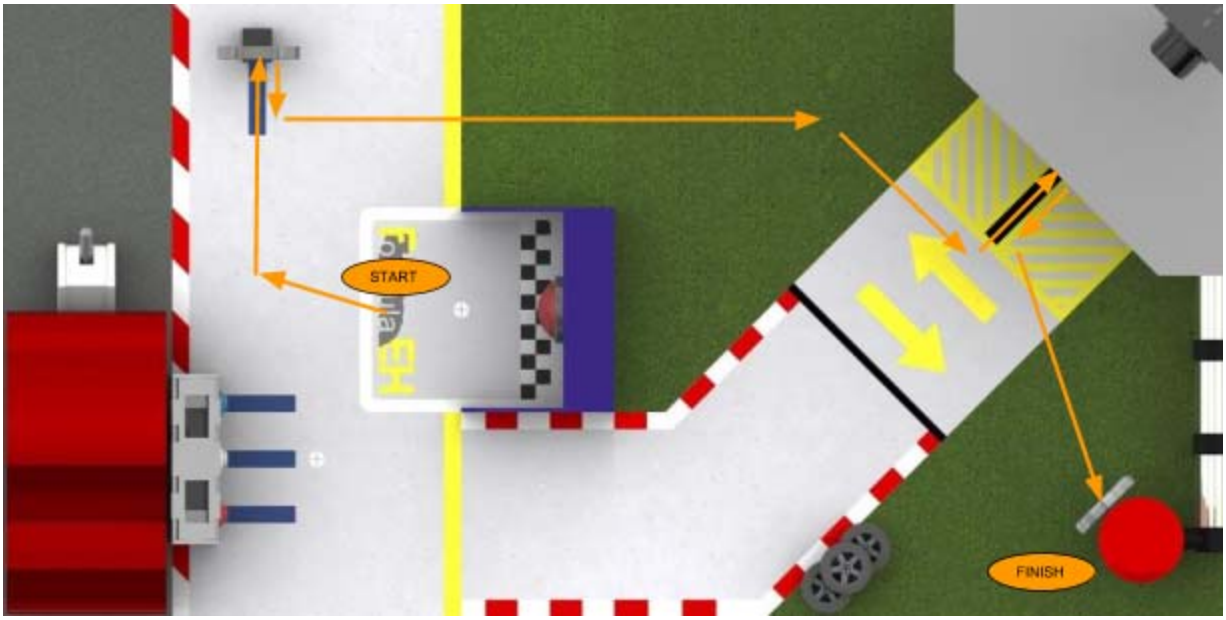


Figure F5: Course strategy for Performance Test 3.

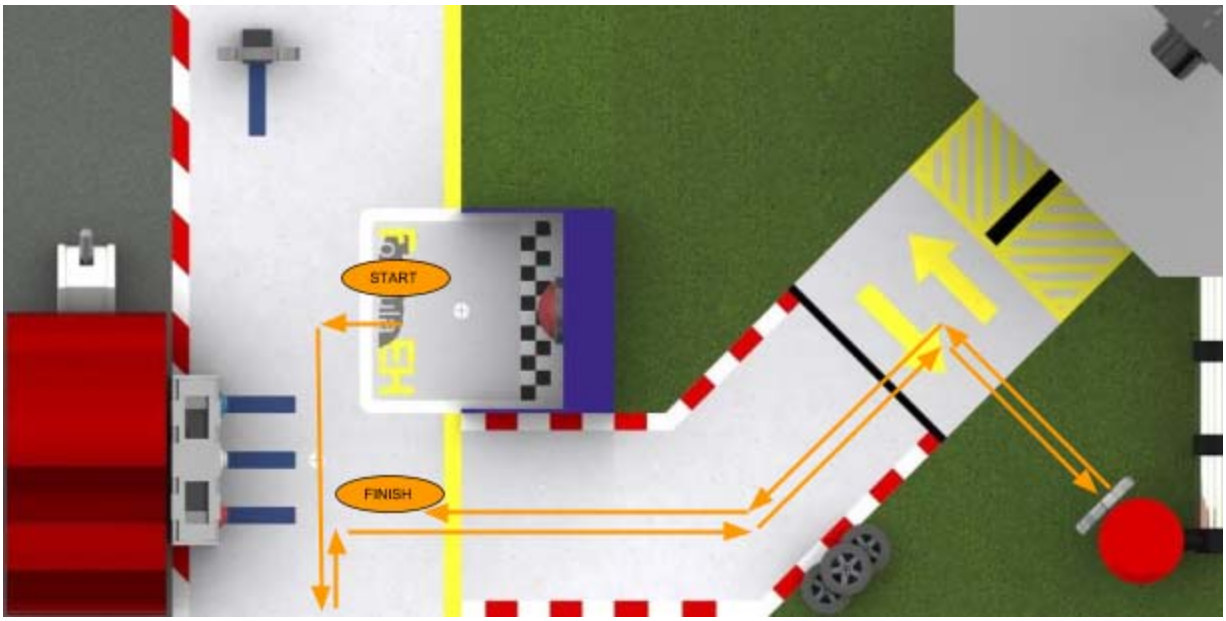


Figure F6: Initial course strategy for Performance Test 4.

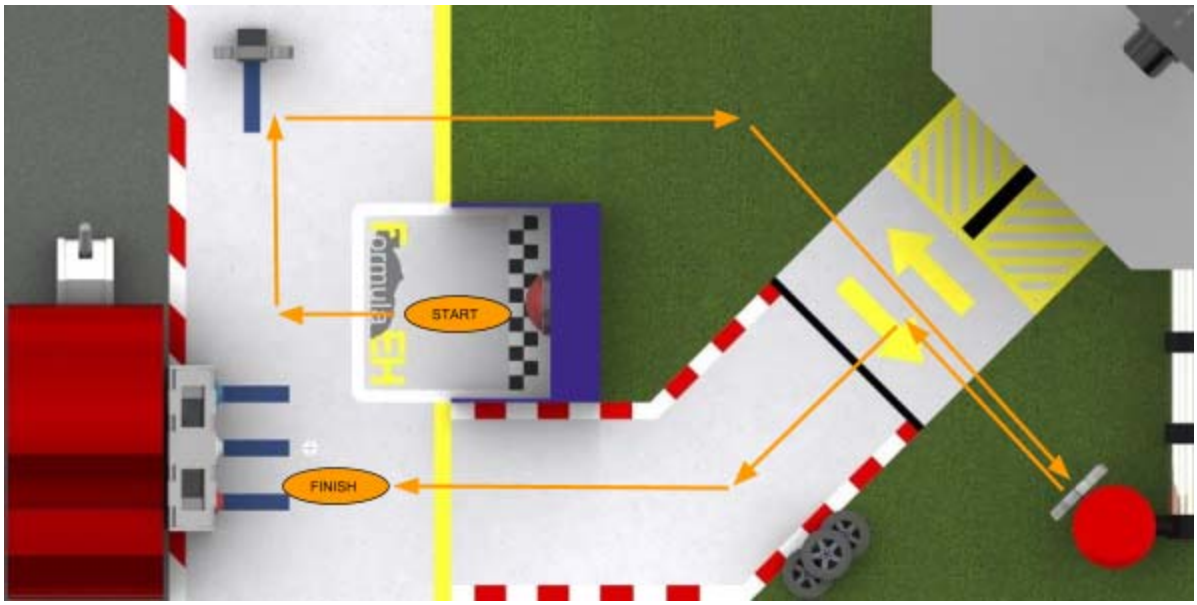


Figure F7: Second course strategy for Performance Test 4.

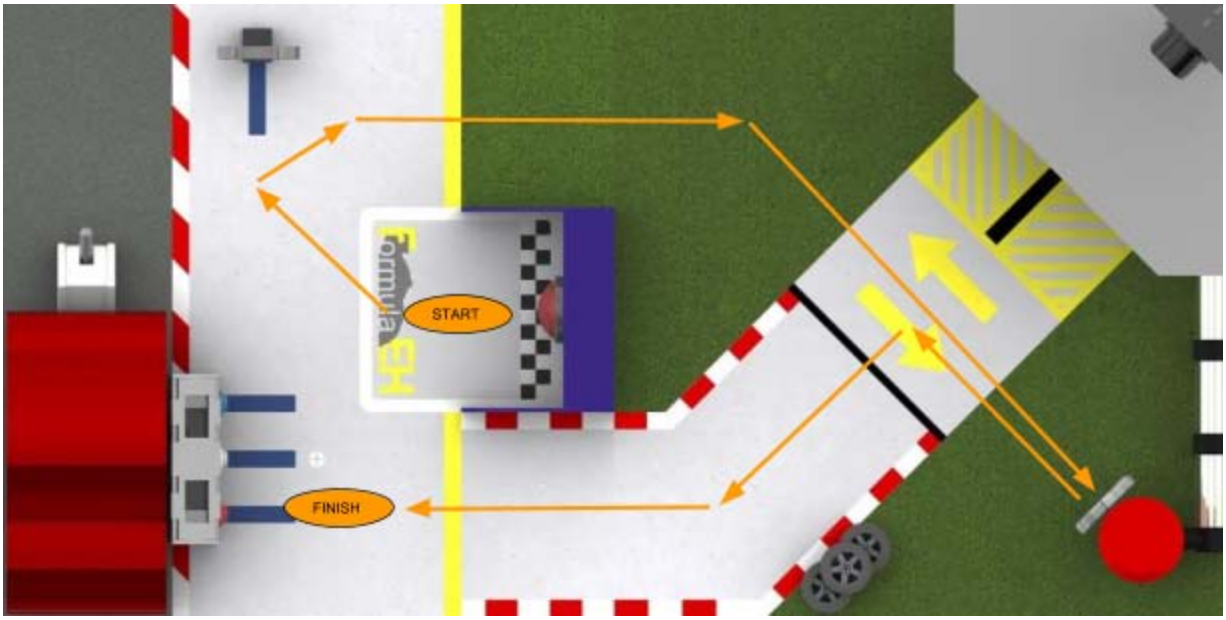


Figure F8: Third course strategy for Performance Test 4.

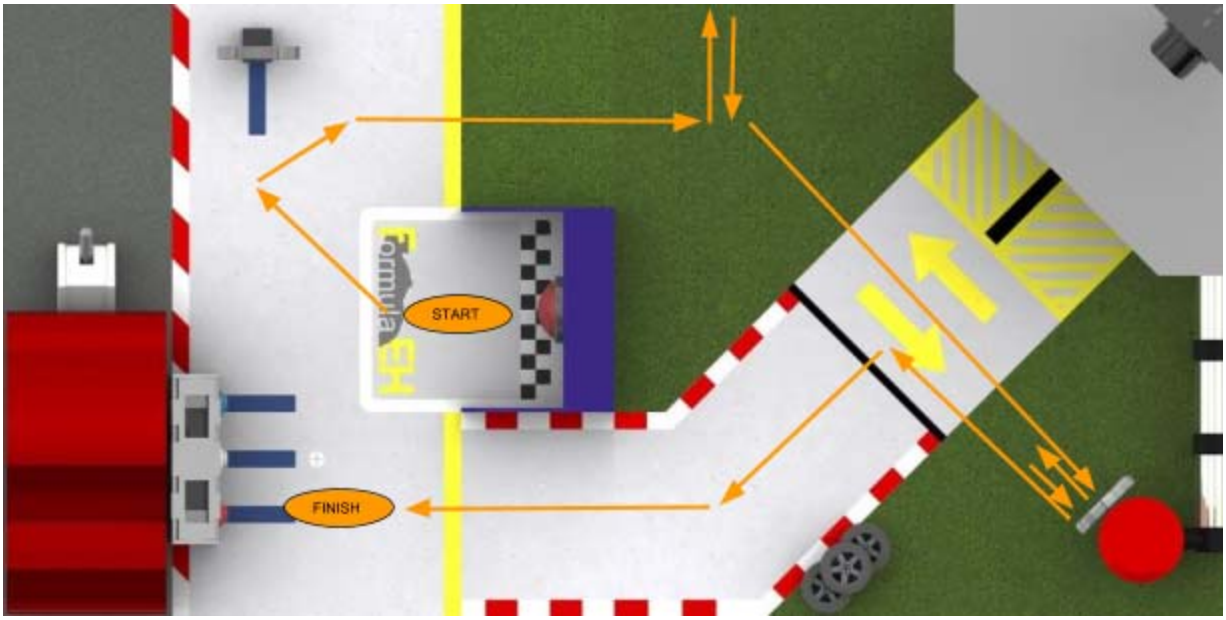


Figure F9: Fourth course strategy for Performance Test 4.

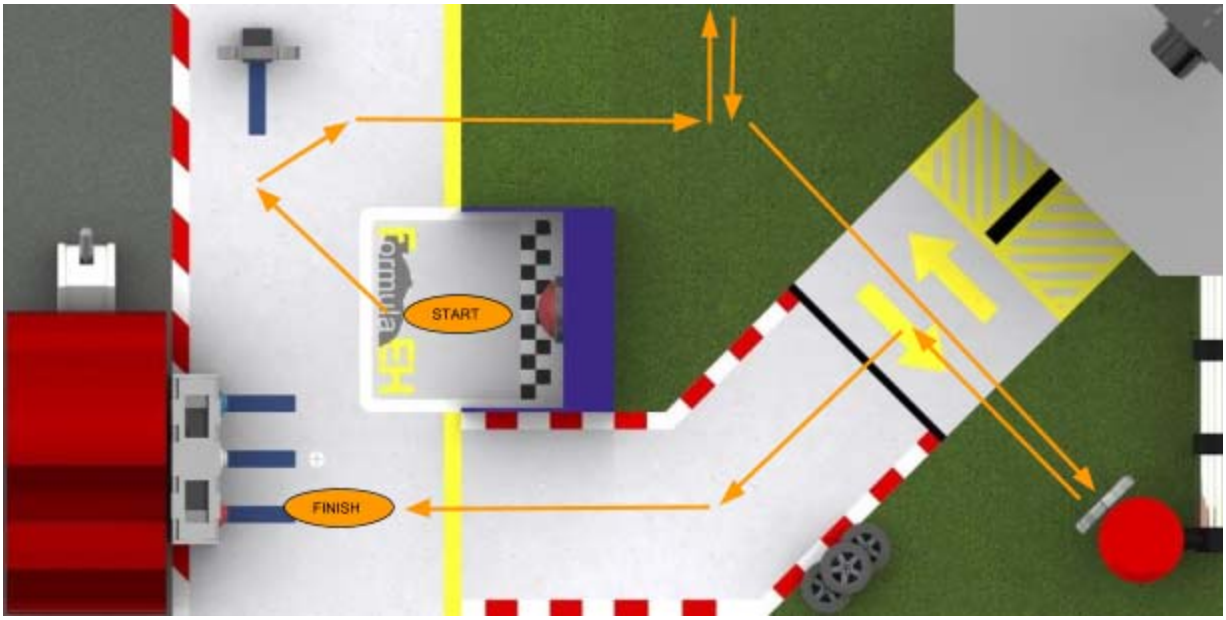


Figure F10: Final course strategy for Performance Test 4.

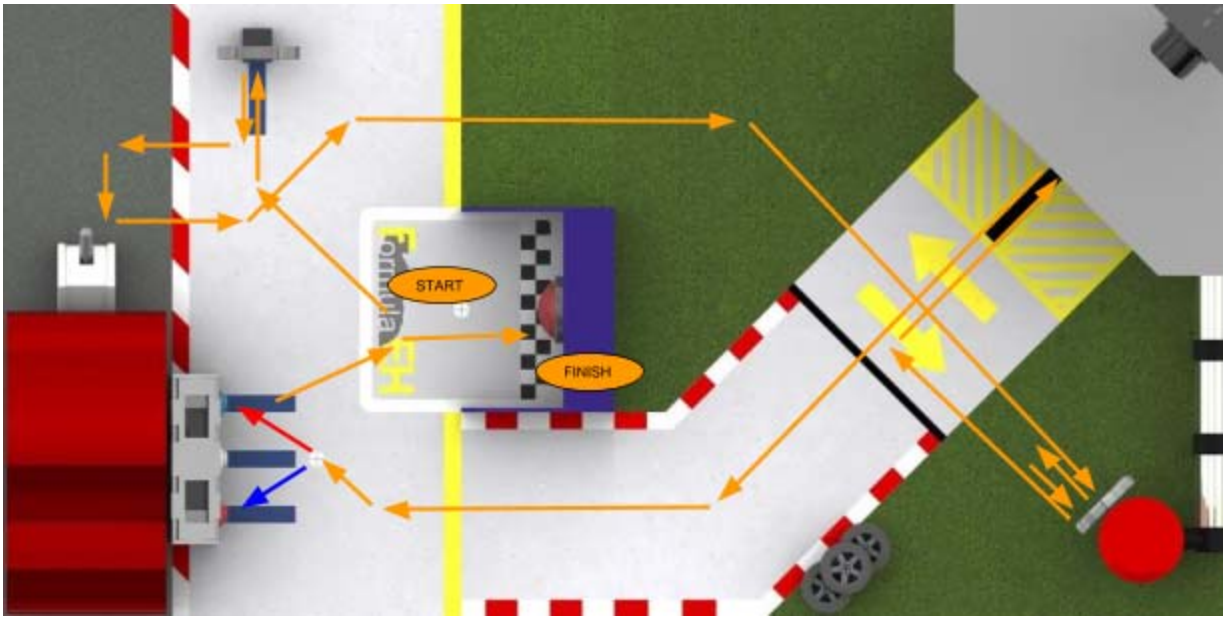


Figure F11: Initial course strategy for the individual competition.

APPENDIX G

Final Design

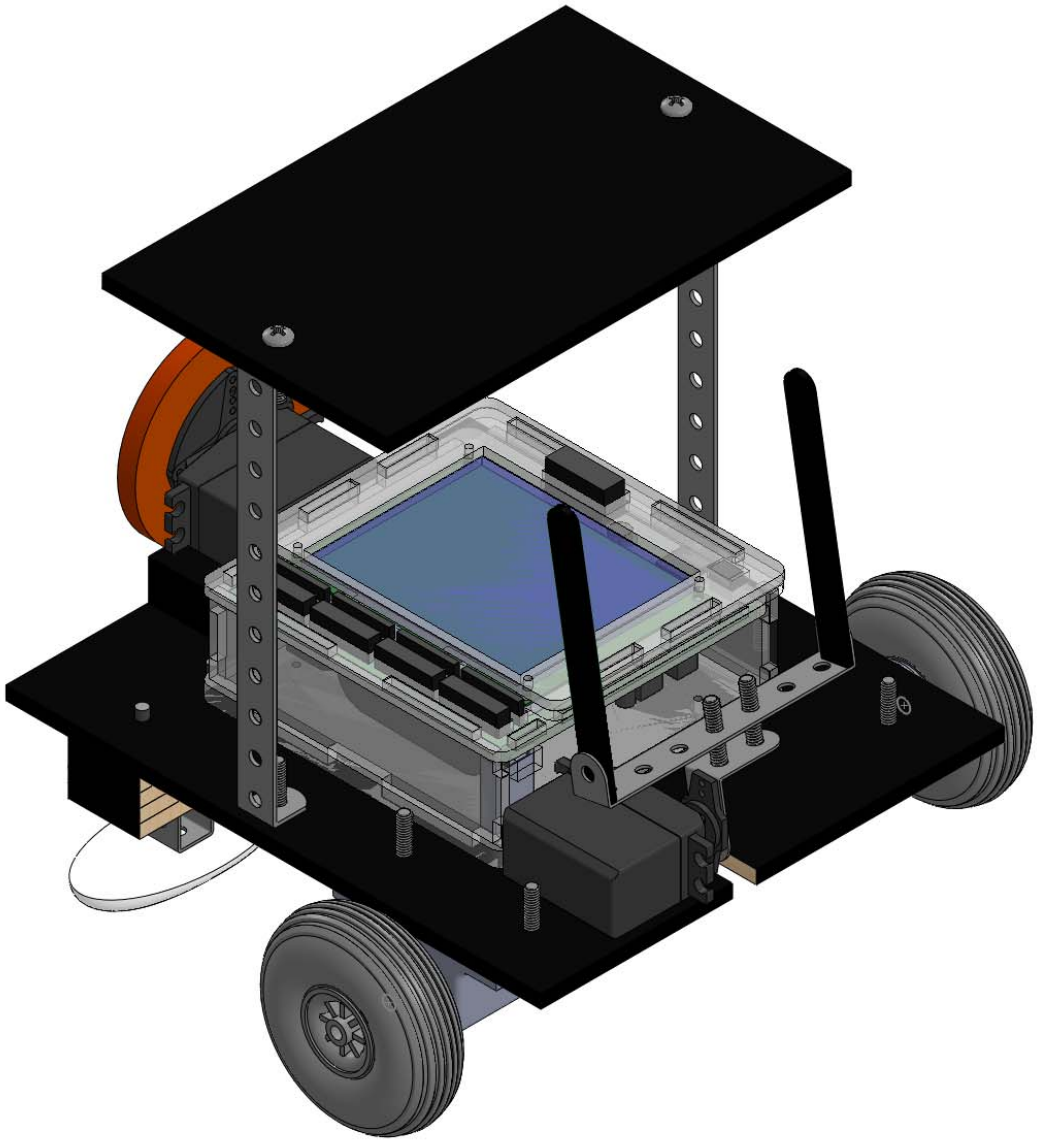
Speed D8ers R19

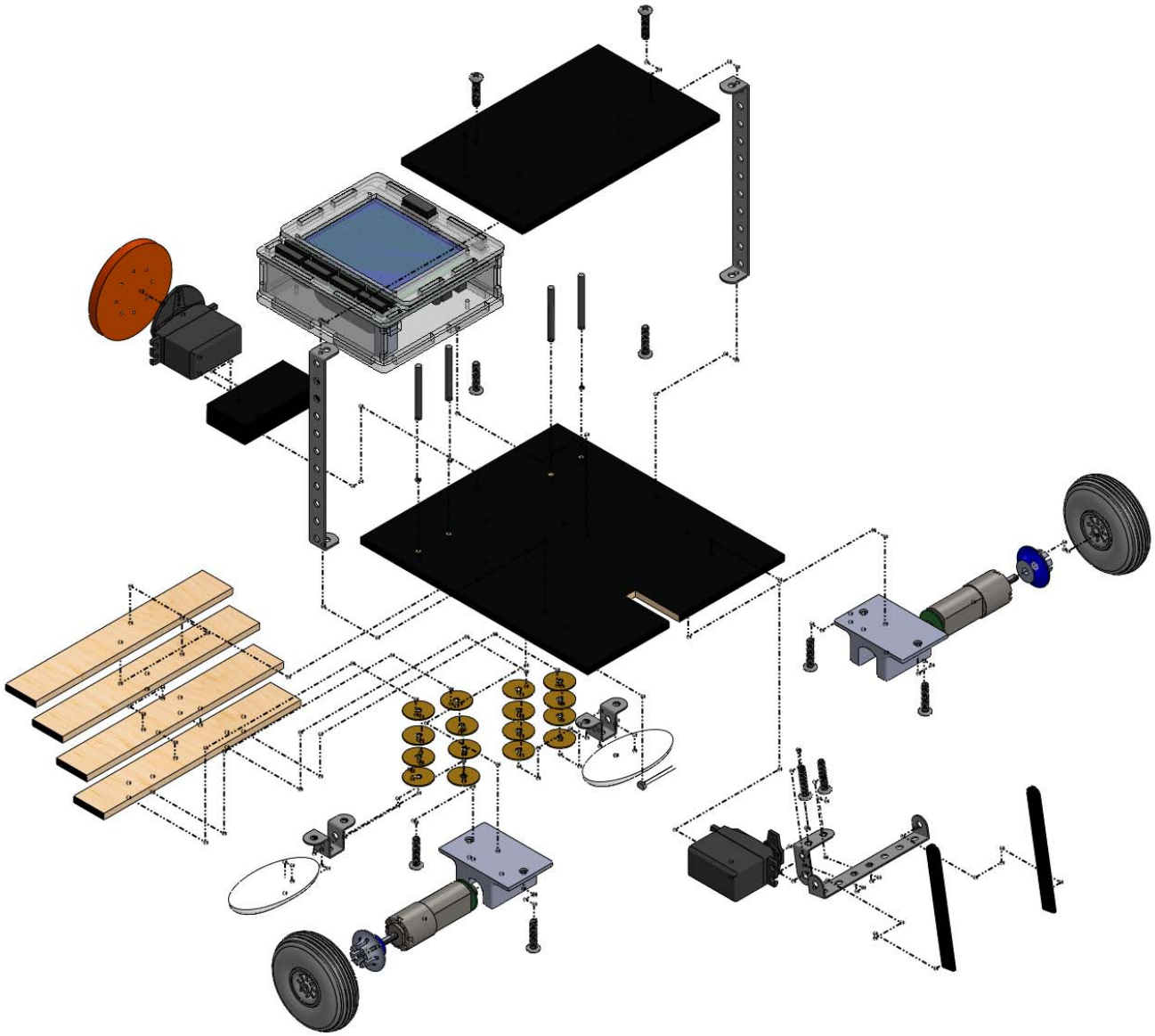
Engineering 1282H
Spring 2018

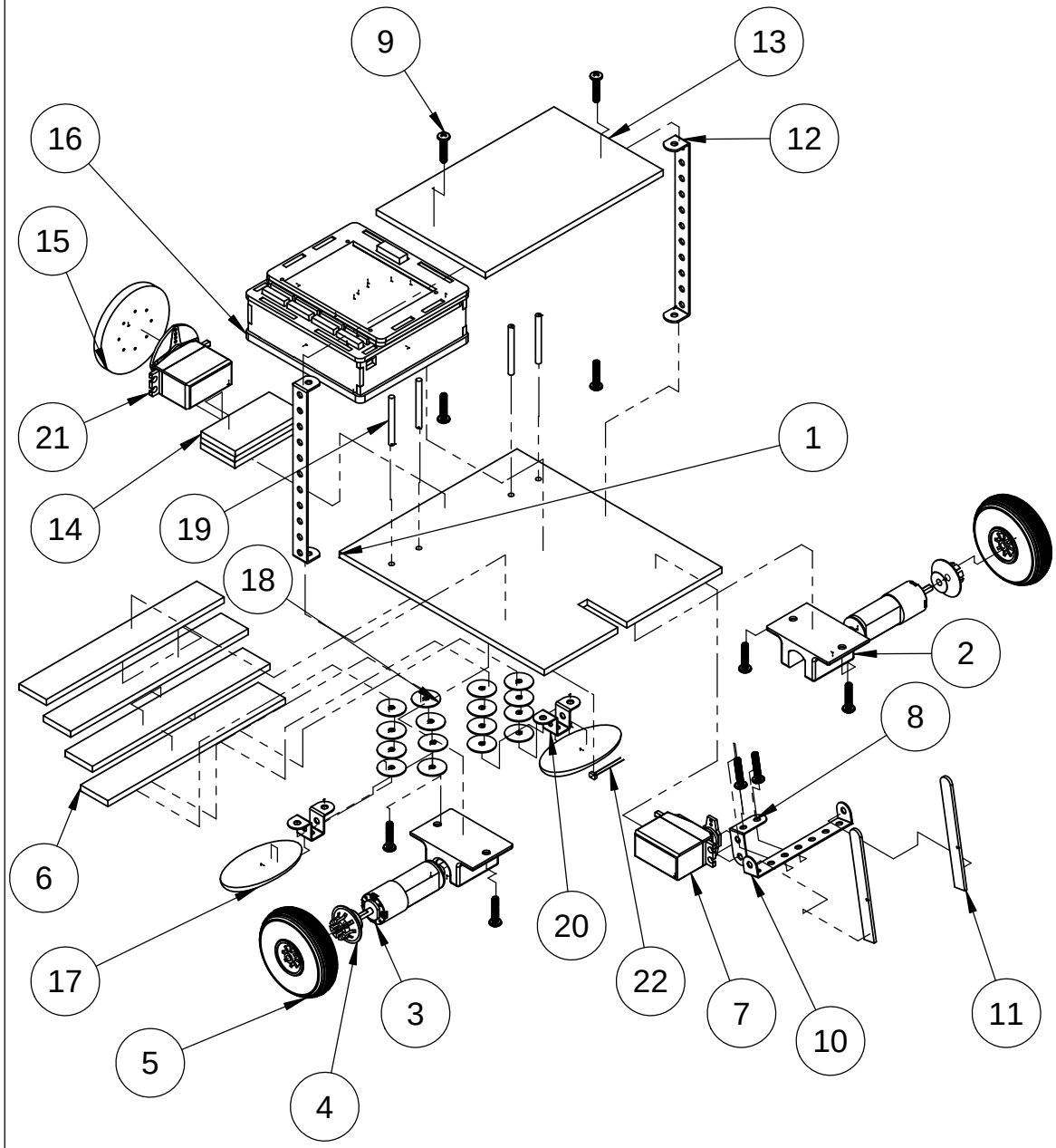
Justin Banke
Jack McGinness
Kelly Meaden
Colby Williams

RJF Mon/Wed/Fri 10:20

Date of Submission: 04/16/18

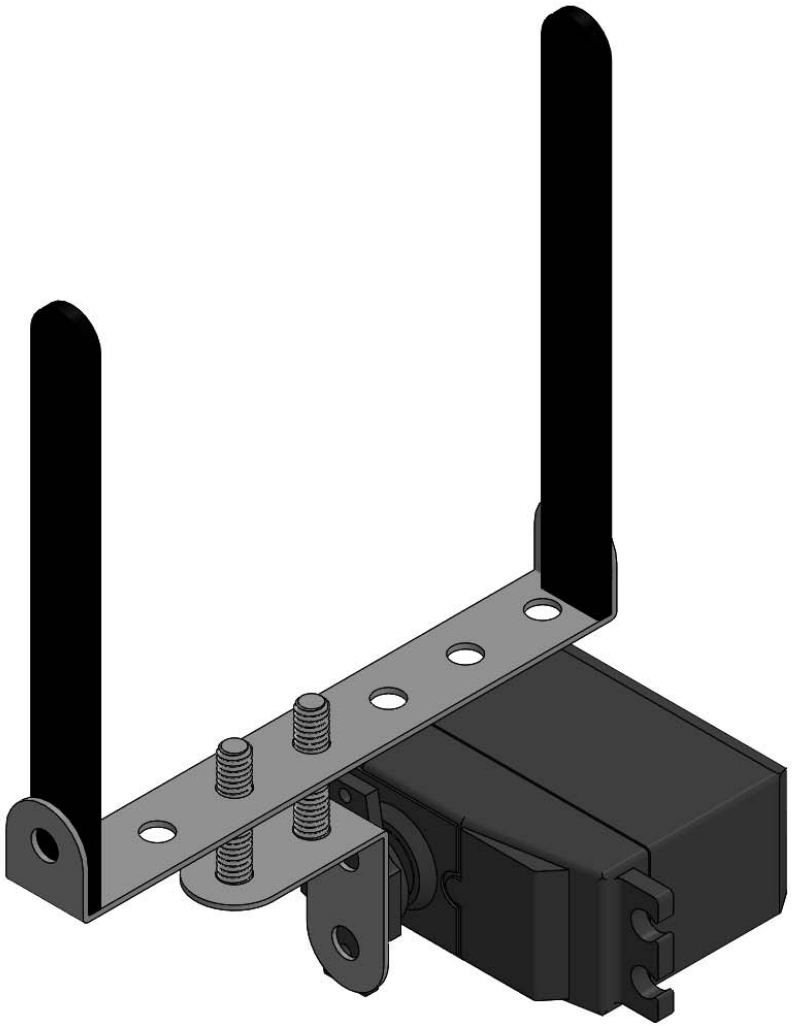


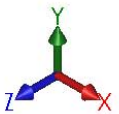


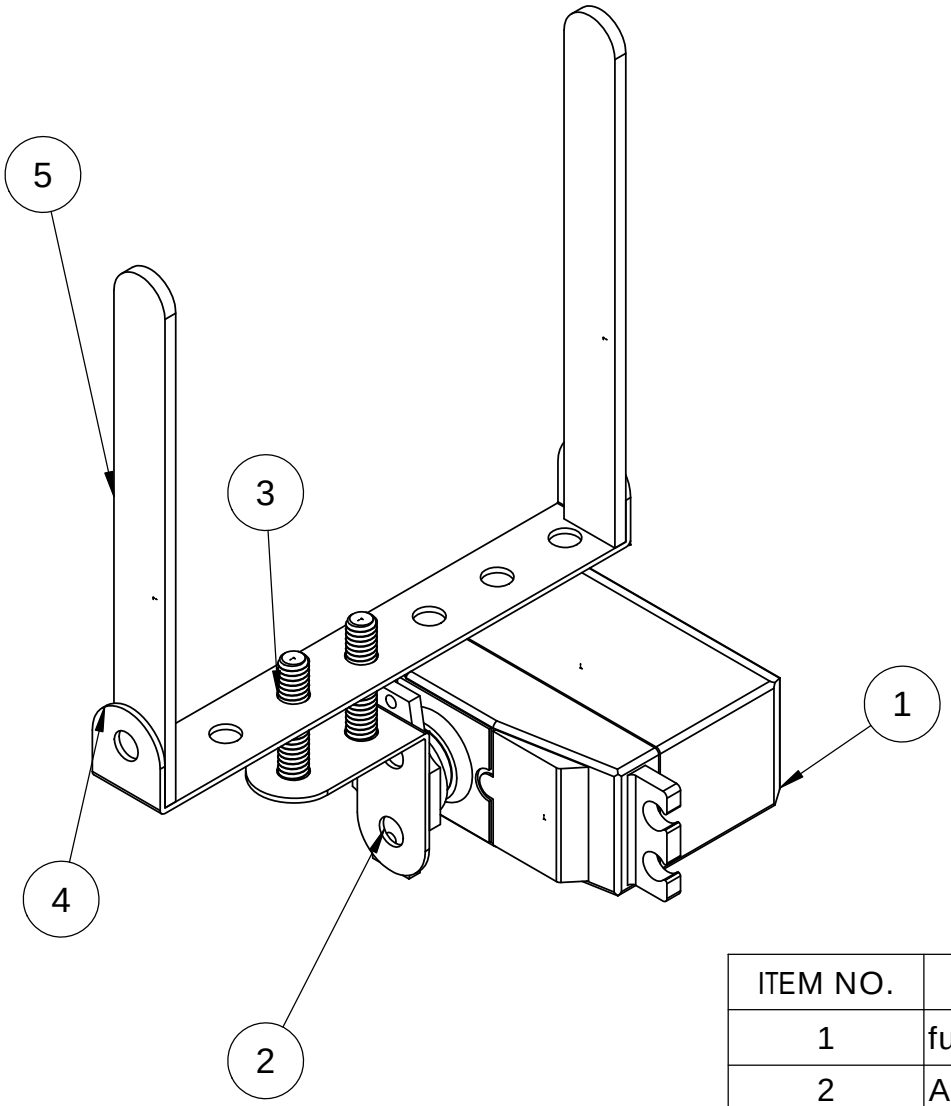


ITEM NO.	PART NAME	MATERIAL	QTY.
1	MDFWoodBody	PLYWOOD	1
2	Igwan Robot Mount	ABS PLASTIC	2
3	IGWAN Motor	STEEL	2
4	IGWAN Wheel Adapter	ABS PLASTIC	2
5	Wheel 2.5"	RUBBER	2
6	MDFWood Skid Stack	PLYWOOD	4
7	futabaServo For Arms	STEEL	1
8	Angle Bracket 2x2	STEEL	1
9	SCREW04	STEEL	10
10	Double Angle Strip 1x7x1	STEEL	1
11	Popsicle Stick	PLYWOOD	2
12	Double Angle Strip 1x11x1	STEEL	2
13	MDFWood3DPrintedMount	PLYWOOD	1
14	MDFCrankMount	PLYWOOD	3
15	3D Printed Crank Wheel	ABS PLASTIC	1
16	PROTEUS	VARIOUS	1
17	Spoon Skids	PLASTIC	2
18	washer	COPPER	16
19	Screwed Rod	STEEL	4
20	Double Bent Strip	STEEL	2
21	futabaServo For Crank	STEEL	1
22	CDS Sensor	STEEL	1

The Ohio State University First Year Engineering	Dwg. Title: Parts List for Final Robot	Inst.: RJF	Scale: 2:9	Dwg. No.: R19
	Drawn By: Team D8 G4	Hour: 10:20	Units: INCH	Date: 4/15/2018

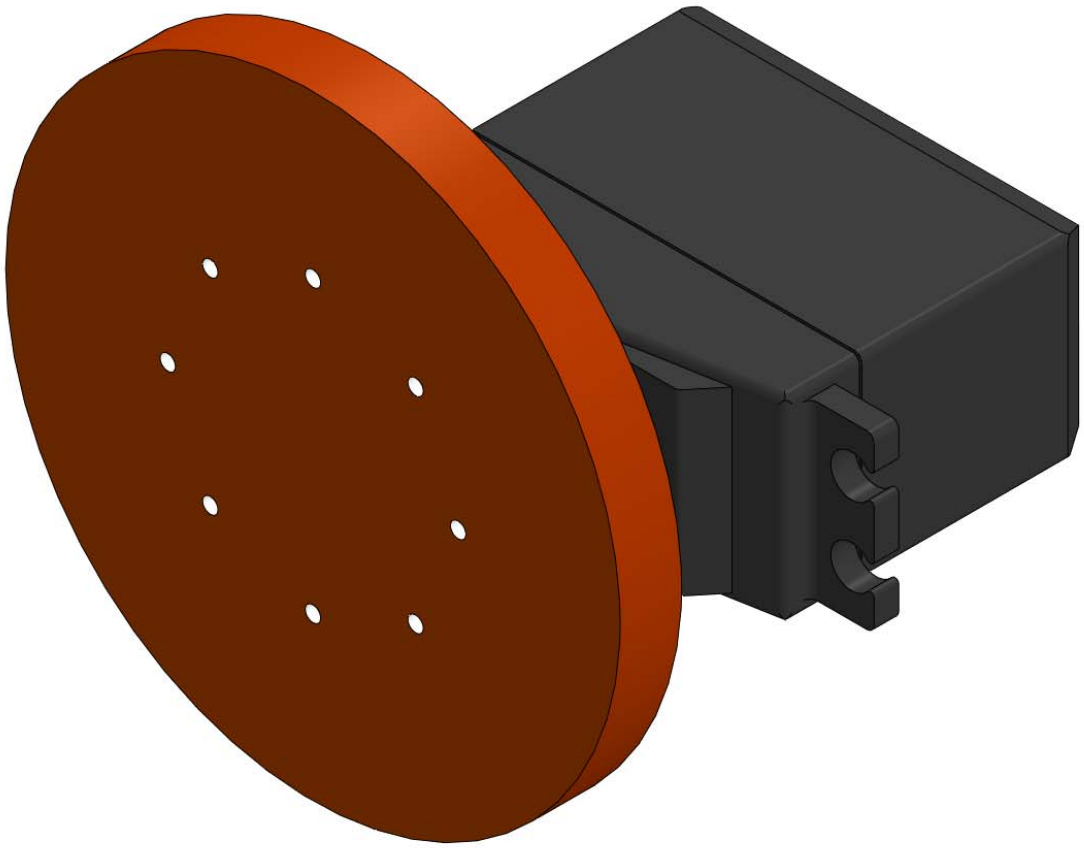


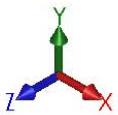

Y
Z
X

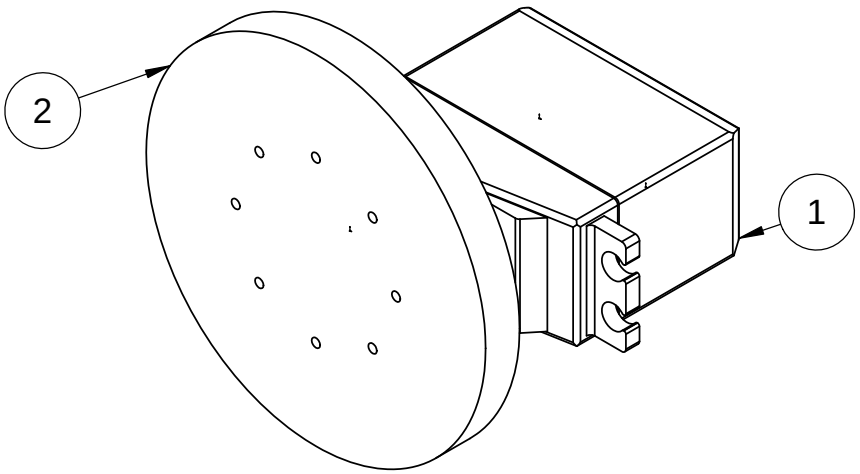


ITEM NO.	PART NUMBER	DESCRIPTION	QTY.
1	futabaServo For Arms	STEEL	1
2	Angle Bracket 2x2	STEEL	1
3	SCREW04	STEEL	2
4	Double Angle Strip 1x7x1	STEEL	1
5	Popsicle Stick	PLYWOOD	2

The Ohio State University First Year Engineering	Dwg. Title: SubAssembly of Wrench Mechanism	Inst.: RJF	Scale: 1:1	Dwg. No.: R19
	Drawn By: Team D8 G6	Hour: 10:20	Units: INCHES	Date: 4/15/2018

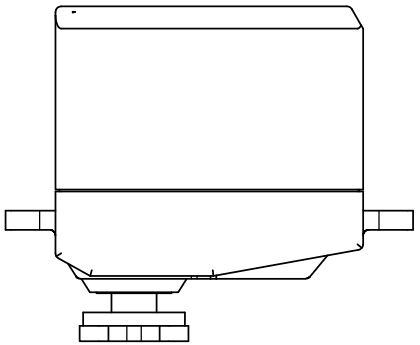


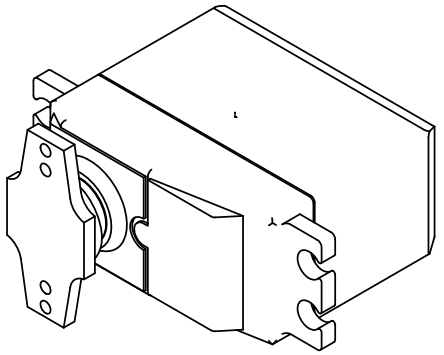

Y
Z
X

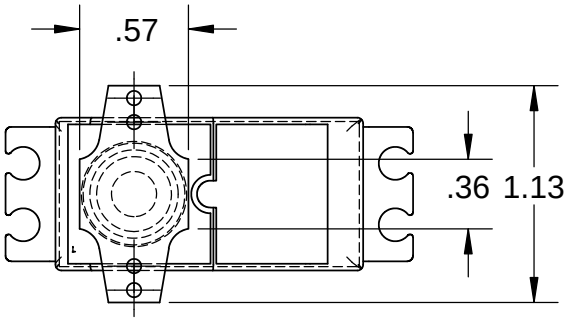


ITEM NO.	PART NUMBER	DESCRIPTION	QTY.
1	futabaServo For Crank	STEEL	1
2	3D Printed Crank Wheel	ABS PLASTIC	1

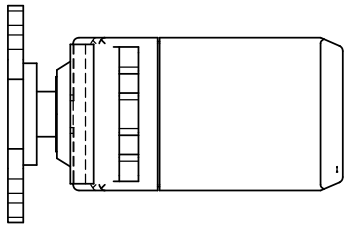
The Ohio State University First Year Engineering	Dwg. Title: SubAssembly of Crank Mechanism	Inst.: RJF	Scale: 1:1	Dwg. No.: R19
	Drawn By: Team D8 G8	Hour: 10:20	Units: INCHES	Date: 4/15/2018



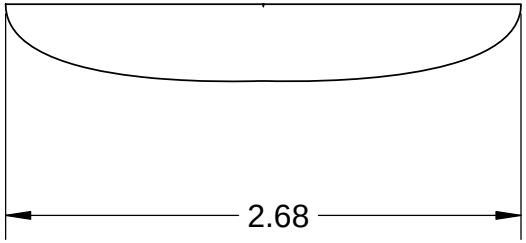


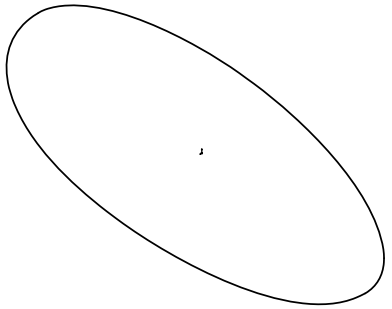


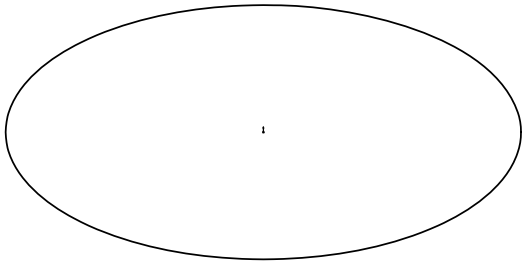
The modification of the servo motor consisted of removing edges from the face of the motor. Cutting off some of the face allowed for better connection, and the abilitity to achieve full range of motion without colliding with main body.

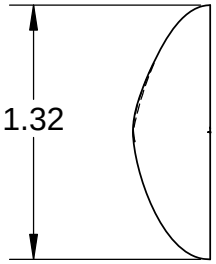


The Ohio State University First Year Engineering	Dwg. Title: Modified Servo Specification	Inst.: RJF	Scale: 1:1	Dwg. No.: R19
	Drawn By: Team D8 G9	Hour: 10:20	Units: INCHES	Date: 4/15/2018

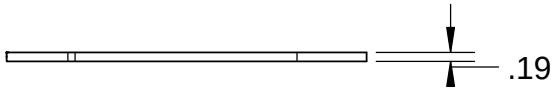


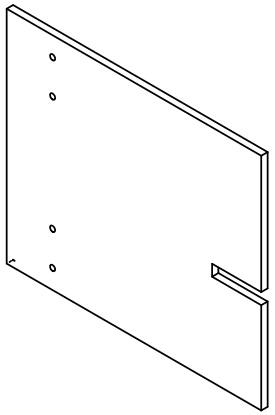


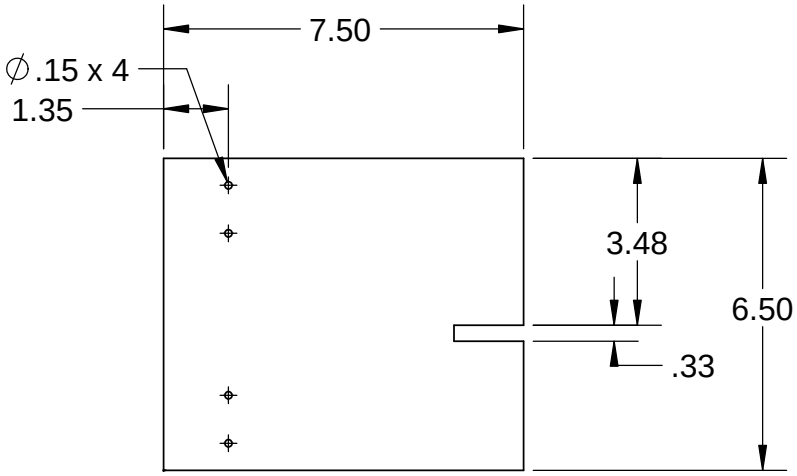


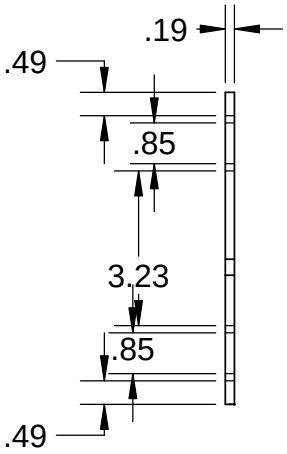


The Ohio State University First Year Engineering	Dwg. Title: Spoon Head Skid	Inst.: RJF	Scale: 1:1	Dwg. No.: R19
	Drawn By: Team D8 G10	Hour: 10:20	Units: INCHES	Date: 4/15/2018

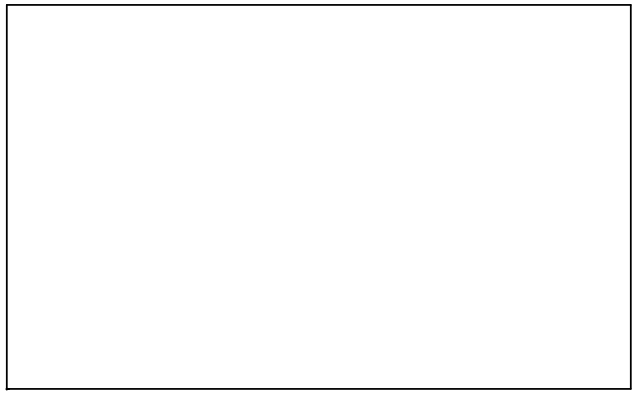


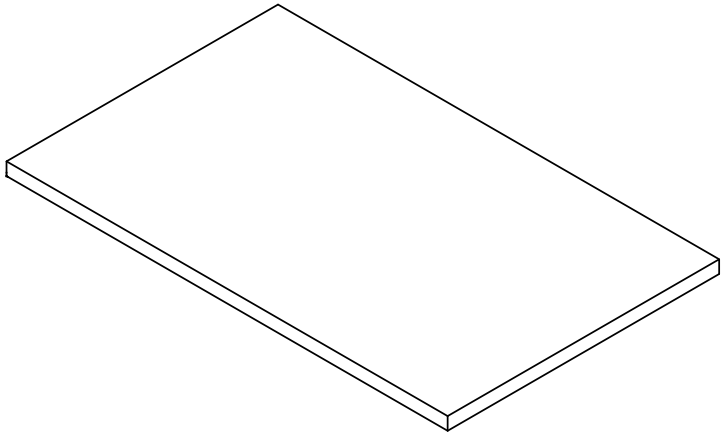


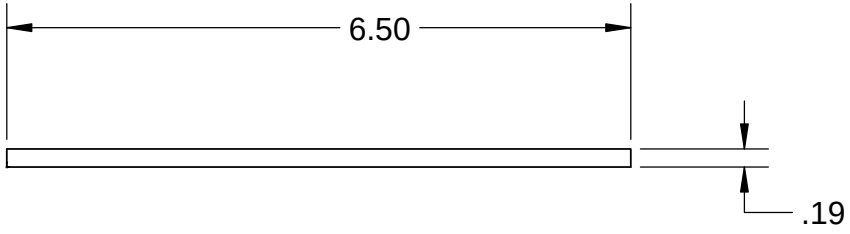


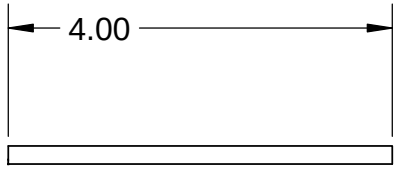


The Ohio State University First Year Engineering	Dwg. Title: Wood Body	Inst.: RJF	Scale: 1:4	Dwg. No.: R19
	Drawn By: Team D8 G11	Hour: 10:20	Units: INCHES	Date: 4/15/18

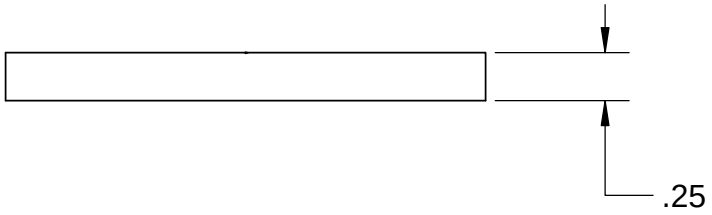


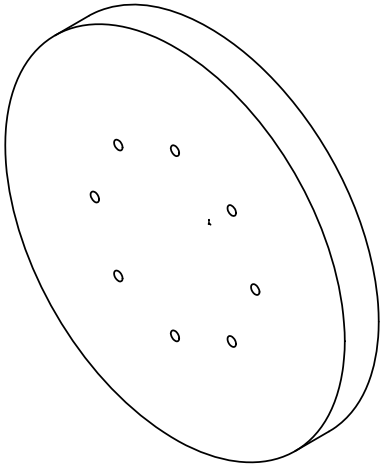


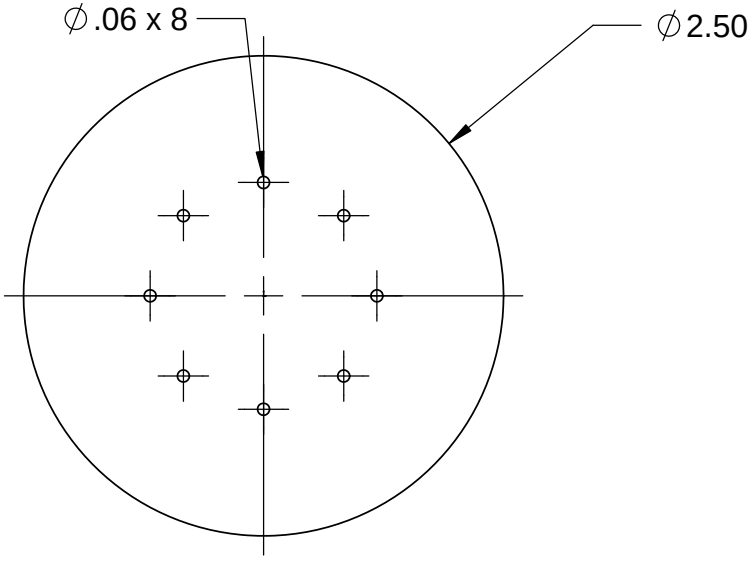




The Ohio State University First Year Engineering	Dwg. Title: QR Mount Wood Part	Inst.: RJF	Scale: 1:2	Dwg. No.: R19
	Drawn By: Team D8 G12	Hour: 10:20	Units: INCHES	Date: 4/15/2018







The Ohio State University First Year Engineering	Dwg. Title: 3D Printed Crank Wheel	Inst.: RJF	Scale: 1:1	Dwg. No.: R19
	Drawn By: Team D8 G13	Hour: 10:20	Units: INCHES	Date: 4/15/2018

APPENDIX H

Electrical Systems

Table H1: Electrical connections table for GPIO ports

Port ID	Wire ID	In-Code Name (Object Name)	Type of Sensor	Purpose	Additional Notes
P0_0					
P0_1	P0_1	lightSensor	CdS Cell	Detects the start light. Identifies button board light	White, yellow, orange wire
P0_2					
P0_3					
P0_4					
P0_5					
P0_6					
P0_7					
P1_0					
P1_1					
P1_2					
P1_3					
P1_4					
P1_5					
P1_6					
P1_7					
P2_0					
P2_1					
P2_2					
P2_3					
P2_4					
P2_5					
P2_6					
P2_7					
P3_0					
P3_1					
P3_2					
P3_3					
P3_4					
P3_5					
P3_6					
P3_7					

Table H2: Electrical connections table for Servo ports

Port ID	Wire ID	Variable Name	Type of Servo	Purpose	Additional Notes
Servo0	S0	fuelServo	Futaba	turn fuel crank	red, white, black wire
Servo1					
Servo2					
Servo3					
Servo4					
Servo5					
Servo6					
Servo7	S1	wrenchServo	Futaba	pick up wrench	red, white, black wire

Table H3: Electrical connections table for Motor ports

Port ID	Wire ID	Variable Name	Type of Motor	Purpose	Additional Notes
Motor0	M0	rightMotor	IGWAN	Drive for right wheel	M0
Motor1	M1	leftMotor	IGWAN	Drive for left wheel	M1
Motor2					
Motor3					

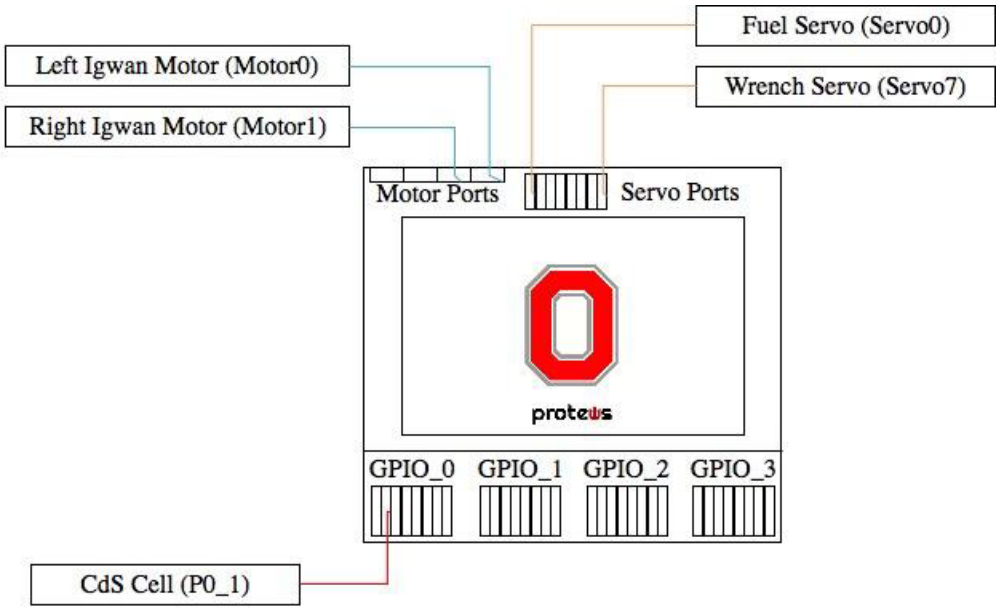


Figure H1: Electrical systems diagram.

H4

APPENDIX I

Code

```
//Includes
#include <FEHLCD.h>
#include <FEHIO.h>
#include <FEHUtility.h>
#include <FEHMotor.h>
#include <FEHServo.h>
#include <FEHRPS.h>

//A ton of defines

//Power
#define POWER 9.0

//Movement Percentages
#define HIGH_FORWARD 50
#define MID_FORWARD 30
#define LOW_FORWARD 20
#define FAST 100

//Turning Percentages (-20/20)
#define INSIDE_TURN_POWER -12
#define OUTSIDE_TURN_POWER 12
#define FAST_INSIDE -50
#define FAST_OUTSIDE 50

//Internal adjustments
#define MOVE_ADJUST2 3
#define TURN_ADJUST 145 //Increase decreases turn angle     317    55.6
#define MOVE_ADJUST 4.6 //Increase increases distance

//Times
#define HALF_SECOND 500
#define FULL_SECOND 1000

//Wrench Servo: 40 degrees is straight up
#define WRENCH_SERVO_MIN 508
#define WRENCH_SERVO_MAX 2061
#define FUEL_SERVO_MIN 500
#define FUEL_SERVO_MAX 2268

//Starting position
#define START_X 16.7
#define START_Y 28.6
```

```
#define START_HEADING 270.0

#define TIMEOUT 10

//Figure out connections
FEHMotor rightMotor(FEHMotor::Motor0,POWER);
FEHMotor leftMotor(FEHMotor::Motor1,POWER);
FEHServo wrenchServo(FEHServo::Servo7);
FEHServo fuelServo(FEHServo::Servo0);
DigitalEncoder rightEncoder(FEHIO::P1_1);
DigitalEncoder leftEncoder(FEHIO::P1_0);
AnalogInputPin lightSensor(FEHIO::P0_1);


int startUp();
void checkMotors();
void showInfo();
void turnRight(float angle);
void moveForward(int percent, float distance);
void readyStart();
float abs(float value);
void runCourse(int test);
bool isBlue();
void pressButton(bool isBlue);
bool ask(char question[]);
void setFuel();
void turnFuel();
void turnTo(float angle);
void moveToX(float x);
void moveToY(float y);

bool isClockwise;

float xCalibrate;
float yCalibrate;
float hCalibrate;


bool testing = true;

//270 points towards -x
//360 points towards +y
//x is jack direction
```

```
int main(void)
{

 int course = startUp();

 if(course != -1) {
    LCD.WriteLine("Ok, I'll run that course!");
    Sleep(FULL_SECOND);
    LCD.WriteLine("Wish me luck!");
    Sleep(FULL_SECOND);

    runCourse(course);

    if(ask("Did I do it right???")) {
       LCD.WriteLine("Yay I did it!");
       Sleep(HALF_SECOND);
       LCD.WriteLine("Was that an official test?");
       Sleep(HALF_SECOND);
       if(ask("Was it?!")) {
          LCD.WriteLine("Yes!  Full points!");
       } else {
          LCD.WriteLine("Lets make it official then");
          Sleep(HALF_SECOND);
          LCD.WriteLine("I got this!");
          Sleep(HALF_SECOND);
          LCD.WriteLine(" ");
          LCD.WriteLine(":)");
       }
    } else {
       LCD.WriteRC("Aw",0,0);
       Sleep(HALF_SECOND);
       LCD.WriteRC(".", 0, 2);
       Sleep(HALF_SECOND);
       LCD.WriteRC(".", 0, 3);
       Sleep(HALF_SECOND);
       LCD.WriteRC(".", 0, 4);
       Sleep(HALF_SECOND);
       LCD.WriteRC("I'll do better next time,",1,0);
       LCD.WriteRC("promise!",2,0);
    }

 } else {
    LCD.WriteLine("Guess I'm done then!");
```

```
    LCD.WriteLine("Good night!");
 }


 return 0;
}

//Runs the specified course
void runCourse(int test) {

 readyStart();

 switch (test) {
 case 0:     //MAIN TEST

    setFuel();

    moveForward(HIGH_FORWARD, 11);
    moveToY(22.5);

    turnRight(-77);
    turnTo(358);

    moveForward(HIGH_FORWARD, 17);
    moveToX(25.5);

    pressButton(isBlue());

    moveForward(HIGH_FORWARD, 13)

    turnRight(-35);

    //THE WRENCH

    wrenchServo.SetDegree(140);         // Set the servos

    moveForward(50, 12);
    moveToY(18.55);
    showInfo();

    // Line up with wrench

    turnRight(30);
```

```
turnTo(180);
showInfo();

// Turn towards wrench

moveForward(LOW_FORWARD, 17);
showInfo();

// Approach wrench

while(RPS.X() > 8.5) {
   moveForward(-1 * LOW_FORWARD, 3);

   if(RPS.Y() > 16.8) {
      turnRight(-5);
   } else {
      turnRight(5);
   }

   moveForward(LOW_FORWARD, 10);

}

wrenchServo.SetDegree(40);
Sleep(HALF_SECOND);

turnTo(180);

moveForward(-50, 6);
showInfo();
            // Back off from wrench

turnRight(-66);
showInfo();
           // Turn to approach jack

moveForward(50, 12);
moveToY(11);
showInfo();
            // Approach jack area

turnRight(-92);
showInfo();
```

```
            // Turn to jack

moveForward(50, 12);
showInfo();
            // Push the jack

moveForward(-50, 5);
showInfo();
            // Back off

turnRight(85);
showInfo();
          // Turn to ramp

moveForward(-50, 15.5);
moveToY(21.5);
showInfo();
            // Move to ramp

turnRight(-42);
turnTo(315);
showInfo();
            // Adjust on ramp

moveForward(-50, 10);
moveToX(4);
showInfo();
            // Move to ramp

turnRight(35);
turnTo(274);
showInfo();
            // Adjust on ramp

bool deadzone;
deadzone = (RPS.IsDeadzoneActive() != 2);


// UP TOP

rightMotor.SetPercent(-12);
leftMotor.SetPercent(-12);
Sleep(0.3);
```

```
rightMotor.SetPercent(-100);
leftMotor.SetPercent(-100);
Sleep(1.2);
rightMotor.SetPercent(-50);
leftMotor.SetPercent(-50);
Sleep(1.05);
rightMotor.Stop();
leftMotor.Stop();


Sleep(300);

moveForward(MID_FORWARD, 30);
moveForward(-1 * MID_FORWARD, 10);
turnRight(-90);
moveForward(-1 * MID_FORWARD, 8);


if(!deadzone) {
    moveToY(46);
}

turnRight(38);
if(!deadzone && RPS.X() != -1) {
    turnTo(226);
}

moveForward(-1 * HIGH_FORWARD, 20);
if(!deadzone && RPS.X() != -1) {
    turnTo(226);
}

moveForward(-1 * HIGH_FORWARD, 20);

moveForward(-1 * HIGH_FORWARD, 10);


turnFuel();                    // FUEL STUFF


moveForward(HIGH_FORWARD, 16);
if(!deadzone && RPS.X() != -1) {
    moveToX(19);
```

```
    }

    turnRight(75);
    if(!deadzone && RPS.X() != -1) {
        turnTo(134);
    }
    moveForward(HIGH_FORWARD, 30);
    moveForward(-1 * MID_FORWARD, 2);
    wrenchServo.SetDegree(150);         // Place wrench in garage
    Sleep(FULL_SECOND);
    Sleep(0.5);
    moveForward(-1 * FAST, 45);         // Go to the ramp again
    wrenchServo.SetDegree(40);
    turnRight(47);                      // Turn to ramp
    moveForward(-1 * FAST, 17);         // Down we go
    turnRight(105);
    moveForward(-1 * HIGH_FORWARD, 20);
    turnRight(105);
    moveForward(-1 * HIGH_FORWARD, 30);

    break;

case 1:     //POSITION BACKWARDS - CHECK 1
    moveForward(-1 * LOW_FORWARD, 8);   // Move out of start
    turnRight(32);
    moveForward(-1 * LOW_FORWARD, 18);  //Move to jack area
    turnRight(42);
    moveForward(-1 * MID_FORWARD, 8);
    moveForward(LOW_FORWARD, 8);        // LEVER
    moveForward(-1 * MID_FORWARD, 10);
    moveForward(LOW_FORWARD, 3);
    turnRight(-45);
    moveForward(LOW_FORWARD, 15);       // Back to center
    turnRight(45);
    moveForward(MID_FORWARD, 20);       // Over to the buttons
    moveForward(-1 * LOW_FORWARD, 4);
    turnRight(90);
    moveForward(LOW_FORWARD, 13);       // Nudge those buttons
    moveForward(-100, 50);              // And away we go
    break;

case 2:      // CHECK 2
```

```
    moveForward(LOW_FORWARD, 6);
    turnRight(85);
    moveForward(LOW_FORWARD, 15);       // Poke the Wrench
    moveForward(-1 * MID_FORWARD, 30);
    moveForward(LOW_FORWARD, 4);        // Ready for buttons
    turnRight(-90);
    moveForward(LOW_FORWARD, 1.5);
    pressButton(isBlue());              // Presses the button
    turnRight(90);
    moveForward(-1 * MID_FORWARD, 10);
    moveForward(LOW_FORWARD, 12);       // Back to the center
    turnRight(90);
    moveForward(LOW_FORWARD, 10);
    moveForward(HIGH_FORWARD, 10);      // Ram the finish button
    break;

case 3:     // CHECK 3
    turnRight(30);
    moveForward(LOW_FORWARD, 18.75);
    turnRight(65);
    wrenchServo.SetDegree(141);
    Sleep(HALF_SECOND);
    moveForward(LOW_FORWARD, 18);       // Approaches wrench
    wrenchServo.SetDegree(40);
    Sleep(HALF_SECOND);
    moveForward(-1 * LOW_FORWARD, 10);   // Backs away with wrench
    turnRight(-135);
    moveForward(-1 * LOW_FORWARD, 17);
    turnRight(35);
    moveForward(-1 * HIGH_FORWARD, 60); // Up the ramp
    turnRight(45);
    moveForward(-1 * LOW_FORWARD, 10);
    turnRight(95);
    moveForward(LOW_FORWARD, 20);
    wrenchServo.SetDegree(150);         // Place wrench in garage
    Sleep(FULL_SECOND);
    moveForward(-1 * LOW_FORWARD, 20);
    turnRight(-90);
    moveForward(-1 * LOW_FORWARD, 30);
    break;

case 4:     // CHECK 4
    setFuel();
```

```
    turnRight(30);
    moveForward(LOW_FORWARD, 18);
    turnRight(-75);
    moveForward(-1 * LOW_FORWARD, 17);
    turnRight(42);
    moveForward(-1 * HIGH_FORWARD, 46); // Up the ramp

    turnRight(110); //straighten on wall
    moveForward(MID_FORWARD, 25);
    leftMotor.SetPercent(HIGH_FORWARD);
    Sleep(FULL_SECOND);
    leftMotor.Stop();
    moveForward(-1 * LOW_FORWARD, 12);

    turnRight(-61);
    moveForward(-1 * LOW_FORWARD, 50);

    turnFuel();                     // Turn the fuel crank
    moveForward(LOW_FORWARD, 18);
    turnRight(-104);
    moveForward(LOW_FORWARD, 31);
    turnRight(49);                  //And right back down
    moveForward(LOW_FORWARD, 30);
 }

 LCD.Clear( FEHLCD::Black );
}

//Run some startup stuff
int startUp() {

 //Check screen
 LCD.Clear( FEHLCD::Black );
 LCD.SetFontColor( FEHLCD::White );
 LCD.WriteLine("Dan, up and running!");
 Sleep(FULL_SECOND);

 wrenchServo.SetMin(WRENCH_SERVO_MIN);
 wrenchServo.SetMax(WRENCH_SERVO_MAX);
 wrenchServo.SetDegree(30);

 fuelServo.SetMin(FUEL_SERVO_MIN);
```

```
fuelServo.SetMax(FUEL_SERVO_MAX);
fuelServo.SetDegree(90);

//Decide what checks to run
if(ask("Should I test my motors?")) {
    checkMotors();
}

RPS.InitializeTouchMenu();

//Decide what course Dan should run
int course;
if (ask("The main test?")) {
    course = 0;
} else if (ask("Should I run PT1?")) {
    course = 1;
} else if (ask("How about PT2?")){
    course = 2;
} else if (ask("PT3 then?")) {
    course = 3;
} else if (ask("Time for PT4?")) {
    course = 4;
} else {
    course = -1;
}

return course;
}

//Checks to see if the motors are functional
void checkMotors() {

//Motor check

LCD.WriteRC("Initiating Motor Check ", 1, 0);
Sleep(HALF_SECOND);
LCD.WriteRC(".", 1, 22);
Sleep(HALF_SECOND);
LCD.WriteRC(".", 1, 23);
Sleep(HALF_SECOND);
LCD.WriteRC(".", 1, 24);
Sleep(HALF_SECOND);
```

```
LCD.Clear( FEHLCD::Black );

LCD.WriteLine("Moving Wrench Servo!");
wrenchServo.SetDegree(100);
Sleep(HALF_SECOND);
wrenchServo.SetDegree(50);
Sleep(FULL_SECOND);

LCD.WriteLine("Moving Fuel Servo!");
fuelServo.SetDegree(100);
Sleep(HALF_SECOND);
fuelServo.SetDegree(50);
Sleep(FULL_SECOND);

LCD.WriteLine("Ok!  Testing Forward!");
rightMotor.SetPercent(LOW_FORWARD);
leftMotor.SetPercent(LOW_FORWARD);
Sleep(FULL_SECOND);
rightMotor.Stop();
leftMotor.Stop();
Sleep(HALF_SECOND);

LCD.WriteLine("Now testing backwards!");
rightMotor.SetPercent(-1 * LOW_FORWARD);
leftMotor.SetPercent(-1 * LOW_FORWARD);
Sleep(FULL_SECOND);
rightMotor.Stop();
leftMotor.Stop();
Sleep(HALF_SECOND);

LCD.WriteLine("Alright, going right now!");
leftMotor.SetPercent(MID_FORWARD);
Sleep(FULL_SECOND);
leftMotor.Stop();
Sleep(HALF_SECOND);

LCD.WriteLine("And finally, left!");
rightMotor.SetPercent(MID_FORWARD);
Sleep(FULL_SECOND);
rightMotor.Stop();
Sleep(HALF_SECOND);
rightMotor.Stop();
```

```
 LCD.Clear( FEHLCD::Black );

 return;

}

//Display the current information from the robot
void showInfo() {

 if(testing) {
 LCD.WriteRC(lightSensor.Value(), 0, 0);
 LCD.WriteRC("X: ", 1, 0);
 LCD.WriteRC(RPS.X(), 1, 3);
 LCD.WriteRC("Y: ", 1, 10);
 LCD.WriteRC(RPS.Y(), 1, 13);
 LCD.WriteRC("Heading: ", 2, 0);
 LCD.WriteRC(RPS.Heading(), 2, 9);
 LCD.WriteRC("Fuel Type (1, Octane): ", 3, 0);
 LCD.WriteRC("Deadzone Active: ", 4, 0);
 }
else {
    LCD.SetFontColor(FEHLCD::White);
    LCD.DrawCircle(159, 119, 59);
    LCD.SetFontColor(FEHLCD::Black);
 }

 return;

}

//Turns to a certain angle.  Negative = left
void turnRight(float angle) {

 //Figure out what direction to turn, set motors
 if(angle < 0) {
    LCD.WriteLine("I'm turning left!");

    rightMotor.SetPercent(OUTSIDE_TURN_POWER);
    leftMotor.SetPercent(INSIDE_TURN_POWER);

    Sleep(100);

    rightMotor.SetPercent(FAST_OUTSIDE);
```

```
    leftMotor.SetPercent(FAST_INSIDE);
 } else {
    LCD.WriteLine("Turning right!");
    rightMotor.SetPercent(FAST_INSIDE);
    leftMotor.SetPercent(FAST_OUTSIDE);
 }

 //Turn based on the angle desired
 Sleep(abs(angle) / TURN_ADJUST);
 rightMotor.Stop();
 leftMotor.Stop();

 return;
}

//Moves forward a certain distance (inch) at the given speed (percent). Negative PERCENT =
backwards
void moveForward(int percent, float distance)
{

if(percent > 0) {
 rightMotor.SetPercent(12);
 leftMotor.SetPercent(12);
} else {
  rightMotor.SetPercent(-12);
  leftMotor.SetPercent(-12);
}

Sleep(100);

//Set both motors to desired percent
rightMotor.SetPercent(percent);
leftMotor.SetPercent(percent);

//Sleep until the distance is reached
Sleep(MOVE_ADJUST2 * distance / abs(percent) - 0.1);

rightMotor.Stop();
leftMotor.Stop();

return;
}
```

```
//Looks for starting light
void readyStart() {

 LCD.WriteLine("I'm in position, ready to start!");
 Sleep(FULL_SECOND);
 LCD.Clear( FEHLCD::Black );

 float timeStart = TimeNow();

 //Display light values as Dan waits for the light to turn on
 while(lightSensor.Value() > 0.9 && timeStart + 30 > TimeNow()) {
    showInfo();
    Sleep(100);
    LCD.Clear( FEHLCD::Black );
 }
 LCD.WriteLine("GO!");


 xCalibrate = START_X - RPS.X();
 yCalibrate = START_Y - RPS.Y();
 hCalibrate = START_HEADING - RPS.Heading();

 return;
}

//Checks light color
bool isBlue() {

 bool isBlue = false;

 //If the light has a value greater than 0.8, it's blue
 if(lightSensor.Value() > 0.8) {
    isBlue = true;
 }

 return isBlue;
}

//Movement to press correct button
void pressButton(bool isBlue) {


 //After reading the color, make corresponding movements
```

```
if(isBlue) {
    LCD.WriteLine("BLUE!");
    LCD.WriteLine(lightSensor.Value());

    //Orient
    moveForward(MID_FORWARD, 1.5);
    turnRight(90);
    turnTo(268);
    rightMotor.SetPercent(12);
    leftMotor.SetPercent(12);
    Sleep(0.3);
    moveForward(FAST, 17);
    moveForward(-1 * LOW_FORWARD, 3);
    rightMotor.SetPercent(100);
    Sleep(6.0);
    rightMotor.Stop();

    //Revert
    moveForward(-1 * MID_FORWARD, 7);
    turnRight(100);
    turnTo(180);
    moveForward(MID_FORWARD, 4.5);


} else {
    LCD.WriteLine("RED!");
    LCD.WriteLine(lightSensor.Value());

    //Orient
    moveForward(-1 * MID_FORWARD, 6);
    turnRight(90);
    turnTo(270);
    //moveForward(MID_FORWARD,15);
    moveForward(FAST, 17);
    moveForward(-1 * LOW_FORWARD, 3);
    leftMotor.SetPercent(100);
    Sleep(6.0);
    leftMotor.Stop();

    //Revert
    moveForward(-1 * MID_FORWARD, 7);
    turnRight(60);
    turnTo(180);
```

```
 }


}

//If it's negative, make it positive
float abs(float value) {

 //If negative, multiply by -1
 if(value < 0) {
    value *= -1;
 }

 return value;
}

//Ask the user a yes/no question
bool ask(char question[]) {

 bool answer = true;

 //Ask the question on screen
 LCD.WriteRC(question, 3, 0);
 LCD.WriteAt("Yes", 70, 130);
 LCD.WriteAt("No", 190, 130);

 float x, y;

 //Wait until the user presses the screen
 while(!LCD.Touch(&x,&y));

 //Take user input, false = no
 if(x > 130) {
    answer = false;
 }
 Sleep(100);

 LCD.Clear( FEHLCD::Black );

 return answer;
}
```

```
//Sets servo angle based on fuel type
void setFuel() {
 if(RPS.FuelType() == 1) {
    // ROTATE CLOCKWISE SETUP
    fuelServo.SetDegree(0);
    isClockwise = true;
 } else {
    // ROTATE COUNTERCLOCKWISE SETUP
    fuelServo.SetDegree(180);
    isClockwise = false;
 }
}

//Turns crank based on fuel type
void turnFuel() {

 if(isClockwise) {

    fuelServo.SetDegree(180);
   moveForward(-1 * LOW_FORWARD,20);
    moveForward(LOW_FORWARD, 3);
    fuelServo.SetDegree(160);
    moveForward(-1 * LOW_FORWARD, 8);
    fuelServo.SetDegree(180);
 } else {
    fuelServo.SetDegree(0);
    moveForward(-1 * LOW_FORWARD,20);
    moveForward(LOW_FORWARD, 3);
    fuelServo.SetDegree(30);
    moveForward(-1 * LOW_FORWARD, 8);
    fuelServo.SetDegree(0);
 }

Sleep(2.0);
}

//Turn to a direction using RPS
void turnTo(float angle) {

 int check = 0;
 bool stop = false;
 float start = TimeNow();
```

```
 while ((RPS.Heading() + hCalibrate  > angle + 1 || RPS.Heading() + hCalibrate < angle - 1) &&
!stop && (start + TIMEOUT > TimeNow())) {
    showInfo();
    if (RPS.Heading() > angle) {



        if ((RPS.Heading() + hCalibrate - angle) > (angle + 360 - RPS.Heading() + hCalibrate)) {
            // TURN LEFT

            if (check == -1) {
                stop = true; //count++
            }

            check = 1;

            rightMotor.SetPercent(OUTSIDE_TURN_POWER);
            leftMotor.SetPercent(INSIDE_TURN_POWER);
        } else {
            // TURN RIGHT

            if (check == 1) {
                stop = true; //count++
            }

            check = -1;


            rightMotor.SetPercent(INSIDE_TURN_POWER);
            leftMotor.SetPercent(OUTSIDE_TURN_POWER);
        }

    } else {


        if ((angle - RPS.Heading() + hCalibrate) > (RPS.Heading() + hCalibrate + 360 - angle)) {
            // TURN RIGHT

            if (check == 1) {
                stop = true; //count++
            }

            check = -1;
```

```
            rightMotor.SetPercent(INSIDE_TURN_POWER);
            leftMotor.SetPercent(OUTSIDE_TURN_POWER);
        } else {
            //TURN LEFT

            if (check == -1) {
                stop = true; //count++
            }

            check = 1;

            rightMotor.SetPercent(OUTSIDE_TURN_POWER);
            leftMotor.SetPercent(INSIDE_TURN_POWER);
        }

    }

    Sleep(100);

    rightMotor.Stop();
    leftMotor.Stop();

    Sleep(100);

 }

 rightMotor.Stop();
 leftMotor.Stop();
}

//Move to an x coord using RPS
void moveToX(float x) {

 int direction = 1;
 int check = 0;
 bool stop = false; //int count = 0;
 float start = TimeNow();

 if(RPS.Heading() > 90 && RPS.Heading() < 270) {
    //Facing the negative x direction
   direction = -1;
```

```
}

 while( (RPS.X() + xCalibrate < x - 0.3 || RPS.X() + xCalibrate > x + 0.3 ) && !stop && (start +
TIMEOUT > TimeNow()))  { //&&count < 3

    if( RPS.X() < x) {
        if (check == -1) {
            stop = true; //count++
        }

        check = 1;
        rightMotor.SetPercent(direction * 12);
        leftMotor.SetPercent(direction * 12);

    } else {
        if (check == 1) {
            stop = true; //count++
        }

        check = -1;
        rightMotor.SetPercent(direction * -12);
        leftMotor.SetPercent(direction * -12);

    }

    Sleep(50);

    rightMotor.Stop();
    leftMotor.Stop();

    Sleep(100);

 }
}

//Move to an y coord using RPS
void moveToY(float y) {

 int direction = 1;
 int check = 0;
 bool stop = false;
 float start = TimeNow();
```

```
 if(RPS.Heading() < 360 && RPS.Heading() > 180) {
     //Facing the negative y direction
    direction = -1;

 }

 while( (RPS.Y() + yCalibrate < y - 0.3 || RPS.Y() + yCalibrate > y + 0.3)  && !stop && (start +
TIMEOUT > TimeNow()))  {

     if( RPS.Y() < y) {

         if (check == -1) {
             stop = true; //count++
         }

         check = 1;

         rightMotor.SetPercent(direction * 12);
         leftMotor.SetPercent(direction * 12);

     } else {

         if (check == 1) {
             stop = true; //count++
         }

         check = -1;

         rightMotor.SetPercent(direction * -12);
         leftMotor.SetPercent(direction * -12);

     }

     Sleep(50);

     rightMotor.Stop();
     leftMotor.Stop();

     Sleep(100);

 }
}
```

APPENDIX J

Code Representation

Algorithm for main:

1. Set variable course to equal the startUp function.
2. Output which test is being run.
3. Run the course using runCourse.
4. Ask user if the test was successful, if yes go to step 5, if no go to step 6.
5. Output “Yay I did it!” then “Was that an official test?” If yes, output “Yes! Full points!”, if no, output “Let’s make it official then. I got this! :)”
6. Output “Aw...I’ll do better next time, promise!”
7. Output “Guess I’m done then! Good night!” once the user is done.

Algorithm for startUp:

1. Output that robot is running.
2. Set the wrenchServo max, min, and degree.
3. Set the fuelServo, min, max, and degree.
4. Ask user if motors should be checked, if yes go to step 5, if no go to step 6.
5. Check that motors are functioning
6. Ask user what course Dan should run. If main test go to step 7, if PT1 go to step 8, if PT2 go to step 9, if PT3 go to step 10, if PT4 go to step 11, and if none of those are selected go to step 12.
7. Return value for main course run, end function.
8. Return value for PT1 run, end function.
9. Return value for PT2 run, end function.
10. Return value for PT3 run, end function.
11. Return value for PT4 run, end function.
12. Return a negative value, end function.

Algorithm for runCourse:

1. If value is 0, run the main course.
2. If value is 1, run the first performance test.
3. If value is 2, run performance test 2.
4. If value is 3, run third performance test.
5. If value is 4, run performance test 4.

Algorithm for ask:

1. Set the answer to true.
2. Write the question on the screen, and an option for yes or no.
3. Wait until user touches the screen.
4. If the user selects no, set answer equal to false.
5. Return answer.

6. End function.

Algorithm for readyStart:

1. Output that Dan is ready to start.
2. Set the start time equal to TimeNow.
3. Display the light values while Dan waits for them to turn on.
4. When light turns on, output GO!.
5. Calibrate the the RPS for x and y as well as the heading.
6. End function.

Algorithm for setFuel:

1. If the RPS checked fuel type is type 1, proceed to step 2. If it is not proceed to step 4.
2. Set the fuel servo to 0 degrees.
3. Set the turn to clockwise.
4. Set the fuel servo to 180 degrees.
5. Set the turn to counterclockwise.
6. End function.

Algorithm for moveForward:

1. Set motor percents to desired value.
2. Sleep until provided distance is reached.
3. Turn off motors.
4. End function.

Algorithm for moveToY:

1. Set the direction to negative, check to 0, stop to false, and start to the current time.
2. If the RPS heading is between 360 and 180 degrees set direction to negative.
3. Begin a loop that runs while the Y location + the yCalibrate are less than y - 0.3, or greater than y + 0.3.
4. If the rps of y is less than variable y proceed to step 5, if not proceed to step 8.
5. If check is -1 set stop equal to true.
6. Set check = 1.
7. Set the right and left motor percents to direction * 12.
8. If the check is equal to 1, set stop equal to true.
9. Set check equal to -1.
10. Set the right and left motor percents to direction * -12.
11. Turn off motors.
12. Exit loop when conditions met.
13. End function.

Algorithm for turnRight:

1. If angle is less than 0, proceed to step 2, otherwise proceed to step 6.
2. Output that the robot is turning left.
3. Set the right motor to the outside turn power, and the left motor to the inside turn power.
4. Pause.
5. Set the right and left motor percents to fast. Proceed to step 8.
6. Output that the robot is turning right.
7. Set the right motor to fast inside, and the left motor to fast outside.
8. Turn based off of the desired angle.
9. Stop the left and right motors.
10. End function.

Algorithm for turnTo:

1. Set check, stop, and start to initial values.
2. Run a loop that runs while the RPS heading + the heading calibration are greater than the angle + 1 or angle -1.
3. If rps heading is greater than the angle proceed to step 4, if not proceed to step 11.
4. If the rps heading + the calibrate - the angle is greater than the angle + 360 proceed to step 5, if not proceed to step 8.
5. If check is equal to -1, set stop equal to true.
6. Set check equal to 1.
7. Set the right motor to the outside power, and the left motor to the inside power. Proceed to step 15.
8. If the check is equal to 1, set stop equal to true.
9. Set check equal to -1.
10. Set the right motor to the inside power, and the left to the outside power. Proceed to step 15.
11. If the angle - rps heading + calibrate is greater than rps heading + calibrate + 360 - angle then proceed to step 12. If not proceed to step 15.
12. If the check is equal to 1, set stop equal to true.
13. Set check equal to -1.
14. Set the right motor to inside turn power, and the left to outside turn power. Go to step 15.
15. Turn off the right and left motors. If loop not broken go back to step 2.
16. Turn off the right and left motors.
17. End function.

Algorithm for moveToX:

1. Set direction equal to 1, check to 0, stop to false, and start to the current time.

2. If the RPS heading is in between 90 and 270 set direction equal to -1.
3. Create a loop that runs while rps x + x calibrate is less than x - 0.3 or greater than x + 0.3.
4. If the rps x is less than variable x proceed to step 5, if not proceed to step 8.
5. If check is equal to -1 set stop equal to true.
6. Set check equal to 1.
7. Set the right and left motor percent to direction * 12. Proceed to step 11.
8. If check is equal to 1 set stop equal to true.
9. Set check equal to -1.
10. Set the right and left motor percents to direction * -12.
11. Turn off the right and left motors. If loop conditions not broken proceed to step 3.
12. End function.

Algorithm for pressButton:

1. Check if the light is blue, if it is proceed to step 2, if it is not proceed to step 5.
2. Orient robot so that it is in front of the blue light.
3. Drive forward and hit the blue light with the chassis.
4. Revert back to previous position. Proceed to step 8.
5. Orient robot so that it is in front of the red light.
6. Drive forward and hit the red light with the chassis.
7. Revert back to previous position.
8. End function.

Algorithm for isBlue:

1. Set if the light is blue to false.
2. Check if the light sensor value is above 0.8, if it is change the light is blue to true.
3. Return light color.
4. End function.

Algorithm for showInfo:

1. Call showInfo and display current information.
2. If the robot is testing, proceed to step 3, if not proceed to step 9.
3. Write the light sensor value to the screen.
4. Write the current X rps coordinate to the screen.
5. Write the current Y rps coordinate to the screen.
6. Display the heading to the screen.
7. Display the fuel type to the screen.
8. Display whether the deadzone is active or not. Proceed to step 10.
9. Draw a circle on the screen.
10. End function.

Algorithm for turnFuel:

1. If turn is clockwise proceed to step 2, if not proceed to step 7.
2. Set the fuel servo degree to 180.
3. Move forward at a low speed.
4. Set the fuel servo degree to 160.
5. Back up and drive forward again.
6. Set the fuel servo degree to 180. Proceed to step 12.
7. Set the fuel servo degree to 0.
8. Move forward.
9. Set the servo degree to 30.
10. Back up and drive forward again.
11. Set fuel servo degree to 0.
12. End function.

Algorithm for checkMotors:

1. Output that motor check is being initiated.
2. Output that wrench servo is being tested.
3. Move wrench servo up and down.
4. Output the fuel servod is being tested.
5. Move fuel servo left and right.
6. Output that the robot will test moving forward.
7. Drive forward slowly.
8. Output that the robot will now test driving backward.
9. Drive backward slowly.
10. Output that the robot will test turning right.
11. Turn right slowly.
12. Output that the robot will test turning left.
13. Turn left slowly.
14. End function.

Algorithm for abs:

1. If the value is less than 0, multiply it by negative 1.
2. Return the value.
3. End function.

APPENDIX K

Budget

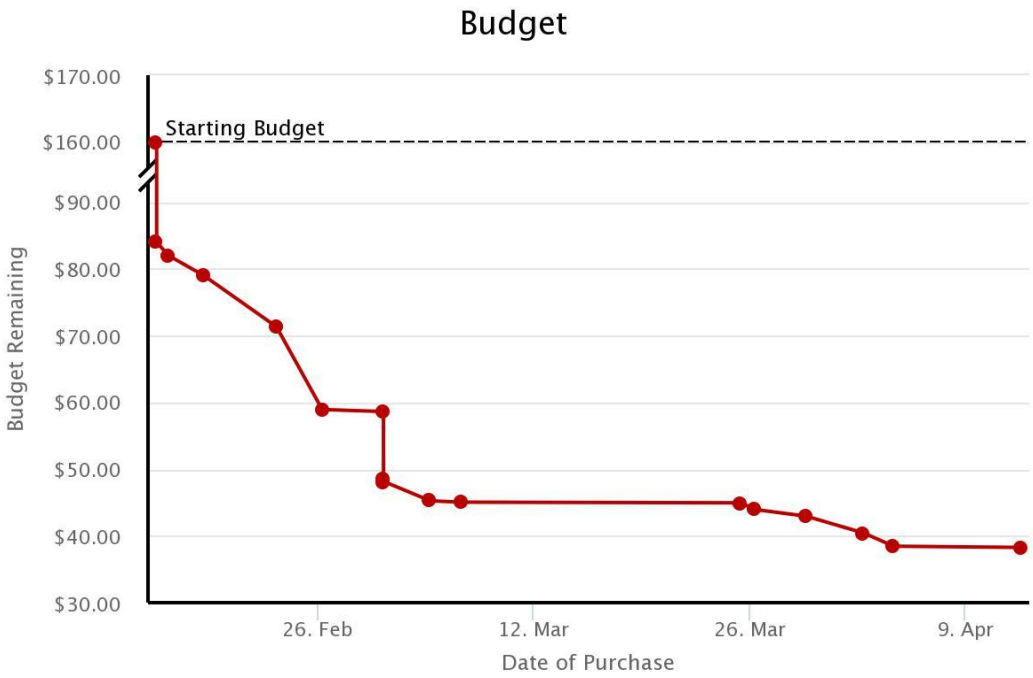


Figure K1: Graph of budget remaining over time [4].

Table K1: List of purchases by item.

Description	Quantity	Cost Per Item
Angle Bracket, 1x1" 2x2 hole	1	$ 1.00
Axle Rod, 8	1	$ 1.48
Washer, .75	8	$ 0.29
Double Bent Strip	2	$ 0.79
Double Angle Strip 3.5"x.5" 1x7x1 hole	1	$ 0.90
Double Angle Strip 5.5"x.5" 1x11x1 hole	2	$ 1.92
Screwed Rod, 3	4	$ 0.96
Fuel Crank Mechanism (3D printed part)	1	$ 2.88
1/4" MDF (Engineered Plywood), 12" x 12" sheet	1	$ 4.30
DuBro Wheel Adapter for IGWAN shaft - only for 2.5 inch wheels	6	$ 1.00
3D printed chassis mount for IGWAN motors	3	$ 2.50
IG-22 (Igwan) motor	2	$ 25.00
Futaba Servo Motor (with Hardware)	2	$ 10.00
Terminal block for continuous rotation motors	2	$ 0.05
DuBro Wheel Adapter, Shaft Mount (with 1 SCREW16) - only for 2.5 inch wheels	2	$ 1.00
POPSICLE STICK OR TONGUE DEPRESSOR	2	$ 0.05
6 Pack #8 Screws (.75"), Nuts, Washers	4	$ 0.20
6 Pack #6 Screws(1.5"), Nuts, Washers	1	$ 0.20
Long Male Header Strip (16 pin)	1	$ 0.50
Velcro strips sold by inch length	2	$ 0.15
Shaft Lock Collar w/set screw (fits erector axle) (with 1 SCREW15)	4	$ 0.35
DuBro 250T 2.5" Wheel (Single)	4	$ 2.60
4" tie wraps (zip ties)	2	$ 0.05
Plastic spoons	2	$ 0.01
Paper	1	$ 0.01
Ping Pong Paddle (2.5" diameter)	1	$ 0.15

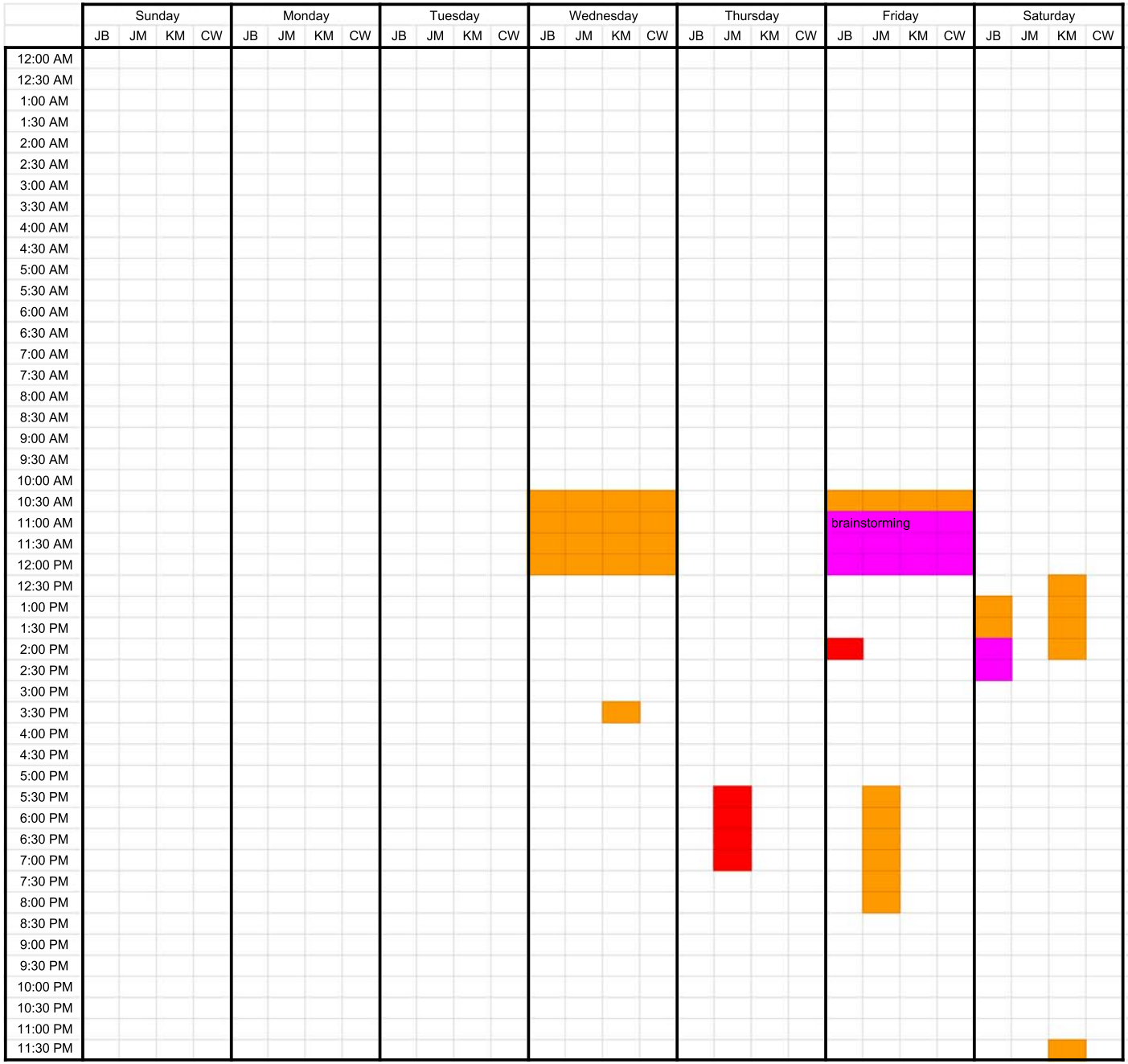
APPENDIX L

Schedule

Table L1: Design schedule.

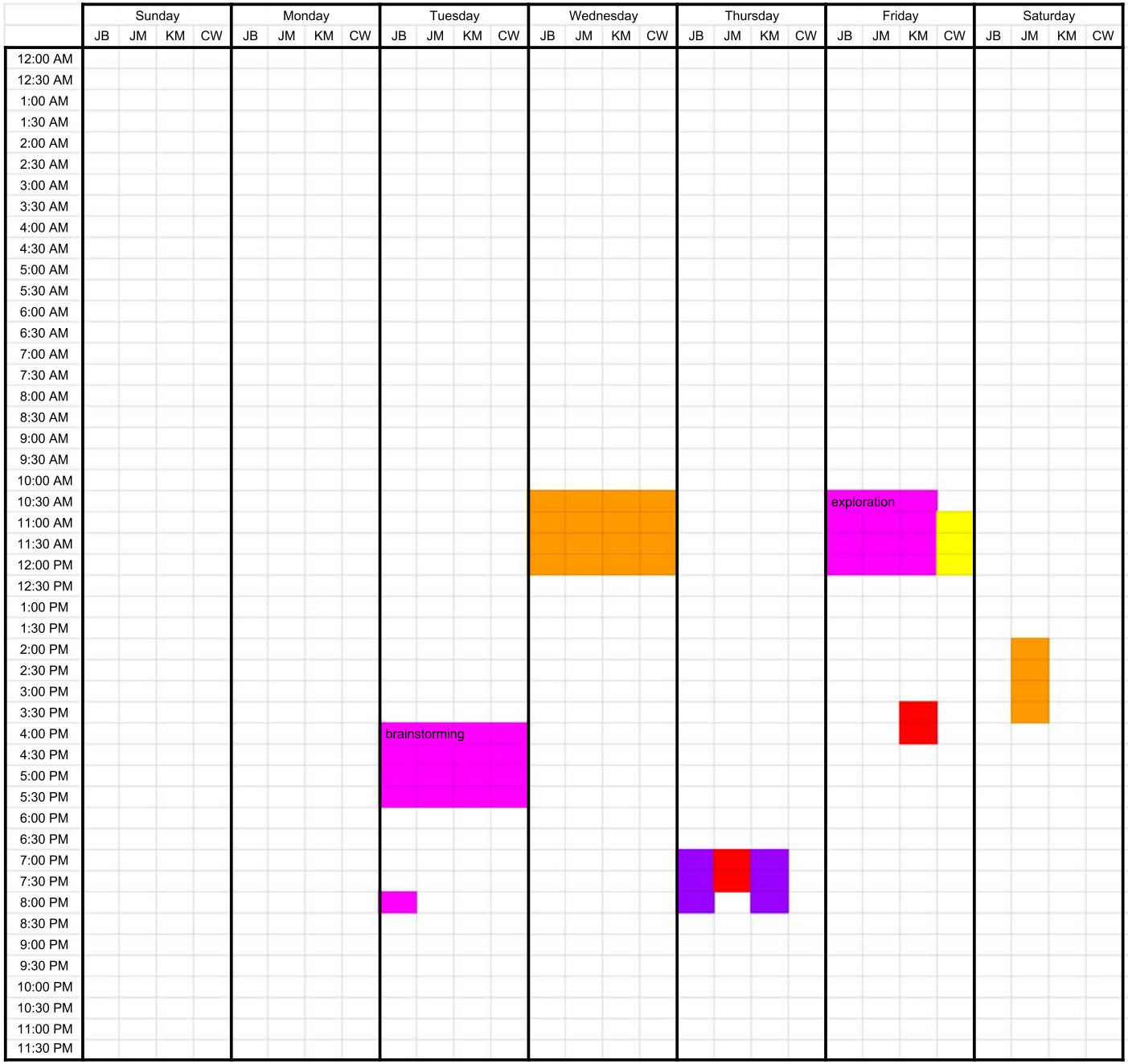
#	DAL Number	Task	Scheduled Start	Scheduled Finish	Actual Start	Actual Finish	Due Date	Colby	Jack	Justin	Kelly	Status	Estimated Time (hrs)	Actual Time (hrs)		
1	R01	**Individual Brainstorming**	1/26/2018	1/30/2018	1/26/2018	1/30/2018	1/31/2018					100%	2	2		Created:
2	R02	**Working Agreement**	1/31/2018	2/1/2018	1/31/2018	2/1/2018	2/2/2018					100%	0.5	0.5		2/3/2018
3		**Team Crest**	1/31/2018	2/2/2018	1/31/2018	1/31/2018	2/2/2018					100%	0.5	0.5		
4	R03	**Group Brainstorming**	1/31/2018	2/6/2018	2/2/2018	2/6/2018	2/7/2018					100%	1.5	2.2		Last Updated:
4A		List Ideas	2/2/2018	2/2/2018	2/2/2018	2/2/2018	2/7/2018					100%	0.25	0.5		4/21/2018
4B		Idea Screening	2/2/2018	2/6/2018	2/2/2018	2/6/2018	2/7/2018					100%	0.25	0.75		2:06 PM
4C		Concept Scoring	2/6/2018	2/6/2018	2/6/2018	2/6/2018	2/7/2018					100%	0.25	0.25		Kelly Meaden
4D		Descriptions and Sketches	2/6/2018	2/6/2018	2/6/2018	2/6/2018	2/7/2018					100%	0.5	0.4		
4E		Strategies	2/6/2018	2/6/2018	2/5/2018	2/6/2018	2/7/2018					100%	0.25	0.3		**Primary**
5	R04	**Design Schedule**	2/2/2018	2/5/2018	2/3/2018	2/7/2018	2/9/2018					100%	1	3		**Secondary**
	R05	**Mockup / Solid Model**	2/2/2018	2/11/2018	2/7/2018	2/11/2018	2/12/2018					100%	3	3		
6		Physical Model	2/2/2018	2/11/2018	2/7/2018	2/8/2018	2/12/2018					100%	2	1.5		
7		SolidWorks Model	2/2/2018	2/11/2018	2/11/2018	2/11/2018	2/12/2018					100%	1	1.5		
8	R06	**Drivetrain Analysis**	2/9/2018	2/13/2018	2/12/2018	2/12/2018	2/14/2018					100%	0.75	0.9		
8A		Part I: Speed	2/9/2018	2/13/2018	2/12/2018	2/12/2018	2/14/2018					100%	0.25	0.3		
8B		Part II: Torque	2/9/2018	2/13/2018	2/12/2018	2/12/2018	2/14/2018					100%	0.25	0.5		
8C		Part III: Motor Selection	2/9/2018	2/13/2018	2/12/2018	2/12/2018	2/14/2018					100%	0.25	0.1		
9	R07	**Exploration 1 Report (Individual)**	2/9/2018	2/15/2018	2/9/2018	2/15/2018	2/16/2018					100%	3	1		
10	R08	**Agendas and Notes**	2/2/2018	2/18/2018	2/2/2018	2/18/2018	2/19/2018					100%	0.55	0.45		
10A		Keep Agenda	2/2/2018	2/18/2018	2/2/2018	2/18/2018	2/19/2018					100%	0.25	0.1		
10B		Take Notes	2/2/2018	2/18/2018	2/2/2018	2/18/2018	2/19/2018					100%	0.25	0.25		
10C		Upload to Project Portfolio	2/2/2018	2/18/2018	2/6/2018	2/18/2018	2/19/2018					100%	0.05	0.1		
11	R09	**Final Report Outline**	2/19/2018	2/20/2018	2/19/2018	2/20/2018	2/21/2018					100%	3	0.5		
12	R10-1	**Report for Peer Feedback**	2/19/2018	2/25/2018	2/23/2018	2/26/2018	2/26/2018					100%	1.05	7.05		
12A		Write Intro & Prelim. Concepts	2/19/2018	2/25/2018	2/23/2018	2/26/2018	2/26/2018					100%	1	7		
12B		Print 4 Copies and Rubric	2/19/2018	2/25/2018	2/26/2018	2/26/2018	2/26/2018					100%	0.05	0.05		
13	R10-2	**Peer Review**	2/26/2018	3/4/2018	2/26/2018	3/5/2018	3/5/2018					100%	1.25	0.9		
13A		Peer Feedback Form	2/26/2018	3/4/2018	2/26/2018	3/2/2018	3/5/2018					100%	0.5	0.25		
13B		Group Discussion	2/26/2018	3/4/2018	3/2/2018	3/2/2018	3/5/2018					100%	0.25	0.15		
13C		Summary	2/26/2018	3/4/2018	3/2/2018	3/5/2018	3/5/2018					100%	0.5	0.5		
		Performance Test 1	2/14/2018	2/22/2018	2/14/2018	2/23/2018	2/23/2018					100%	8	9		
14		Navigation to Car Jack	2/14/2018	2/22/2018	2/14/2018	2/23/2018	2/23/2018					100%	4	7		
15		Car Jack Flip	2/14/2018	2/22/2018	2/14/2018	2/23/2018	2/23/2018					100%	2	1		
16		Move to Top Level	2/14/2018	2/22/2018	2/15/2018	2/23/2018	2/23/2018					100%	2	1		
17	R11	**Exploration 2 Report (Individual)**	2/16/2018	2/22/2018	2/16/2018	2/22/2018	2/23/2018					100%	4	1.5		
18	R12	**Code Representation**	2/21/2018	2/25/2018	2/23/2018	2/25/2018	2/26/2018					100%	13	8.5		
18A		Write Code	2/21/2018	2/25/2018	2/10/2018	2/25/2018	2/26/2018					100%	10	7		
18B		Represent Code Logic	2/21/2018	2/25/2018	2/23/2018	2/25/2018	2/26/2018					100%	2	1		
18C		Describe Code and Logic	2/21/2018	2/25/2018	2/25/2018	2/25/2018	2/26/2018					100%	1	0.5		
19	R13	**Budget and Testing Log**	2/10/2018	2/17/2018	2/10/2018	2/17/2018	2/18/2018					100%	2.25	3		
19A		Budget Summary	2/10/2018	2/17/2018	2/10/2018	2/17/2018	2/18/2018					100%	1	1		
19B		Time Summary	2/10/2018	2/17/2018	2/17/2018	2/17/2018	2/18/2018					100%	1	1		
19C		Sample Page of Testing Log	2/10/2018	2/17/2018	2/17/2018	2/17/2018	2/18/2018					100%	0.25	1		
		Performance Test 2	2/22/2018	3/1/2018	2/22/2018	2/23/2018	3/2/2018					100%	6	0.75		
20		Navigate to Button Panel	2/22/2018	3/1/2018	2/22/2018	2/23/2018	3/2/2018					100%	2	0.25		
21		Read Light	2/22/2018	3/1/2018	2/22/2018	2/23/2018	3/2/2018					100%	2	0.25		
22		Push Correct Button	2/22/2018	3/1/2018	2/22/2018	2/23/2018	3/2/2018					100%	2	0.25		
23	R14	**Exploration 3 Report (Individual)**	2/28/2018	3/6/2018	2/28/2018	3/7/2018	3/7/2018					100%	4	1.5		

#	DAL Number	Task	Scheduled Start	Scheduled Finish	Actual Start	Actual Finish	Due Date	Colby	Jack	Justin	Kelly	Status	Estimated Time (hrs)	Actual Time (hrs)
		Performance Test 3	3/2/2018	3/8/2018	2/28/2018	3/2/2018	3/9/2018					100%	7	1.75
24		Navigate to Wrench	3/2/2018	3/8/2018	2/28/2018	3/2/2018	3/9/2018					100%	2	0.25
25		Remove Wrench from Holder	3/2/2018	3/8/2018	2/28/2018	3/2/2018	3/9/2018					100%	2	0.5
26		Deposit Wrench in Garage	3/2/2018	3/8/2018	2/28/2018	3/2/2018	3/9/2018					100%	3	1
27	R15	**Final Report First Draft**	3/5/2018	3/20/2018	2/19/2018	3/20/2018	3/21/2018					100%	5	10
		Performance Test 4	3/9/2018	3/22/2018	3/2/2018	3/23/2018	3/23/2018					100%	5	12
28		Navigate to Fuel Pump	3/9/2018	3/22/2018	3/2/2018	3/23/2018	3/23/2018					100%	2	8
29		Turn Crank 180 Degrees	3/9/2018	3/22/2018	3/2/2018	3/23/2018	3/23/2018					100%	3	4
30	R16	**Electrical Systems**	3/7/2018	3/25/2018	3/21/2018	3/21/2018	3/26/2018					100%	4	0.5
30A		Diagram	3/7/2018	3/25/2018	3/21/2018	3/21/2018	3/26/2018					100%	2	0.25
30B		Tables	3/7/2018	3/25/2018	3/21/2018	3/21/2018	3/26/2018					100%	2	0.25
31	R17	**Final Report Second Draft**	3/21/2018	4/1/2018	3/20/2018	4/1/2018	4/2/2018					100%	5	8
32	R18	**Isometric for Display**	3/30/2018	4/3/2018	3/17/2018	3/28/2018	4/4/2018					100%	3.25	6.15
32A		CAD Drawing	3/30/2018	4/3/2018	3/17/2018	3/28/2018	4/4/2018					100%	3	6
32B		Qt Creator Installation	3/30/2018	4/3/2018	12/5/2017	12/5/2017	4/4/2018					100%	0.25	0.15
33	R19	**Working Drawing Set**	3/23/2018	4/15/2018	3/23/2018	4/15/2018	4/16/2018					100%	14	16
33A		Non-exploded Assembly	3/23/2018	4/15/2018	3/23/2018	4/15/2018	4/16/2018					100%	2	5
33B		Exploded Assembly	3/23/2018	4/15/2018	3/23/2018	4/15/2018	4/16/2018					100%	3	4
33C		Subassemblies	3/23/2018	4/15/2018	3/23/2018	4/15/2018	4/16/2018					100%	2	2
33D		Bill of Materials	3/23/2018	4/15/2018	4/13/2018	4/15/2018	4/16/2018					100%	1	1
33E		Detail Drawings	3/23/2018	4/15/2018	4/13/2018	4/15/2018	4/16/2018					100%	6	4
34	R20	**Oral Report**	3/21/2018	4/17/2018	4/14/2018	4/17/2018	4/18/2018					100%	2	4
35	R21	**Final Written Report**	2/19/2018	4/22/2018	2/19/2018	4/22/2018	4/23/2018					99%	7	8
36	R22	**Project Portfolio**	1/31/2018	4/22/2018	2/6/2018	4/22/2018	4/23/2018					100%	1.85	11.2
36A		Navigation Menu	1/31/2018	4/22/2018	2/8/2018	4/22/2018	4/23/2018					100%	0.25	0.25
36B		Organizational Pages	1/31/2018	4/22/2018	2/8/2018	4/22/2018	4/23/2018					100%	0.25	3
36C		Team Contact Info	1/31/2018	4/22/2018	2/6/2018	2/6/2018	4/23/2018					100%	0.1	0.05
36D		Design Schedule	1/31/2018	4/22/2018	2/19/2018	2/19/2018	4/23/2018					100%	0.1	0.05
36E		Budget	1/31/2018	4/22/2018	2/19/2018	4/22/2018	4/23/2018					100%	0.1	0.25
36F		Agendas and Notes	1/31/2018	4/22/2018	2/6/2018	4/22/2018	4/23/2018					100%	0.25	2
36G		Sketches, Models, Drawings	1/31/2018	4/22/2018	4/21/2018	4/22/2018	4/23/2018					100%	0.25	0.5
36H		Electrical Systems	1/31/2018	4/22/2018	4/21/2018	4/22/2018	4/23/2018					100%	0.1	0.25
36I		Code	1/31/2018	4/22/2018	4/3/2018	4/22/2018	4/23/2018					100%	0.1	2
36J		Code Representation	1/31/2018	4/22/2018	4/21/2018	4/22/2018	4/23/2018					100%	0.1	2
36K		Testing Logs	1/31/2018	4/22/2018	2/21/2018	4/22/2018	4/23/2018					100%	0.1	0.25
36L		Testing Results	1/31/2018	4/22/2018	3/19/2018	4/22/2018	4/23/2018					100%	0.1	0.5
36M		References	1/31/2018	4/22/2018	4/21/2018	4/22/2018	4/23/2018					100%	0.05	0.1



Week: Jan. 27 - Feb. 3

Category	Total	JB	JM	KM	CW
Documentation	2.5	0.5	2	0	0
Project Management	16	3.5	5	5	2.5
Coding	0	0	0	0	0
Testing	0	0	0	0	0
CAD	0	0	0	0	0
Building	0	0	0	0	0
Other	7	2.5	1.5	1.5	1.5
Total Hours:	25.5	6.5	8.5	6.5	4



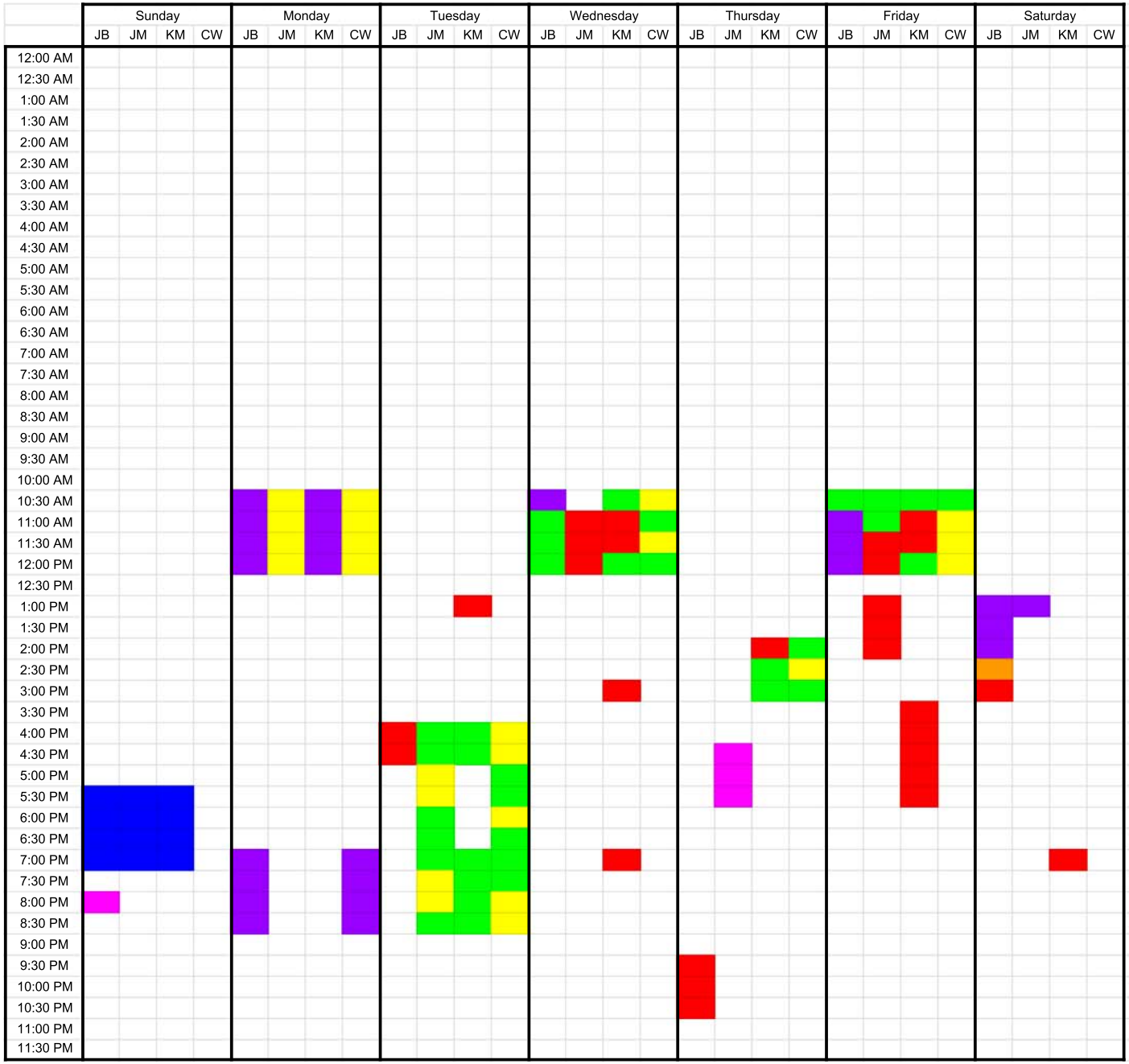
Week: Feb. 4 - Feb. 10

Category	Total	JB	JM	KM	CW
Documentation	2	0	1	1	0
Project Management	10	2	4	2	2
Coding	1.5	0		0	1.5
Testing	0	0		0	0
CAD	0	0		0	0
Building	3	1.5		1.5	0
Other	14.5	4.5	4	4	2
Total Hours:	31	8	9	8.5	5.5

	Sunday				Monday				Tuesday				Wednesday				Thursday				Friday				Saturday			
	JB	JM	KM	CW	JB	JM	KM	CW	JB	JM	KM	CW	JB	JM	KM	CW	JB	JM	KM	CW	JB	JM	KM	CW	JB	JM	KM	CW
12:00 AM																												
12:30 AM																												
1:00 AM																												
1:30 AM																												
2:00 AM																												
2:30 AM																												
3:00 AM																												
3:30 AM																												
4:00 AM																												
4:30 AM																												
5:00 AM																												
5:30 AM																												
6:00 AM																												
6:30 AM																												
7:00 AM																												
7:30 AM																												
8:00 AM																												
8:30 AM																												
9:00 AM																												
9:30 AM																												
10:00 AM																												
10:30 AM					brainstorming								brainstorming								exploration							
11:00 AM																												
11:30 AM					drivetrain calculations																							
12:00 PM																												
12:30 PM																												
1:00 PM																												
1:30 PM																												
2:00 PM																												
2:30 PM																												
3:00 PM																												
3:30 PM																												
4:00 PM																												
4:30 PM																	brainstorming											
5:00 PM																												
5:30 PM																												
6:00 PM																												
6:30 PM																												
7:00 PM																												
7:30 PM																												
8:00 PM																												
8:30 PM																												
9:00 PM																												
9:30 PM																												
10:00 PM																												
10:30 PM																												
11:00 PM																												
11:30 PM																												

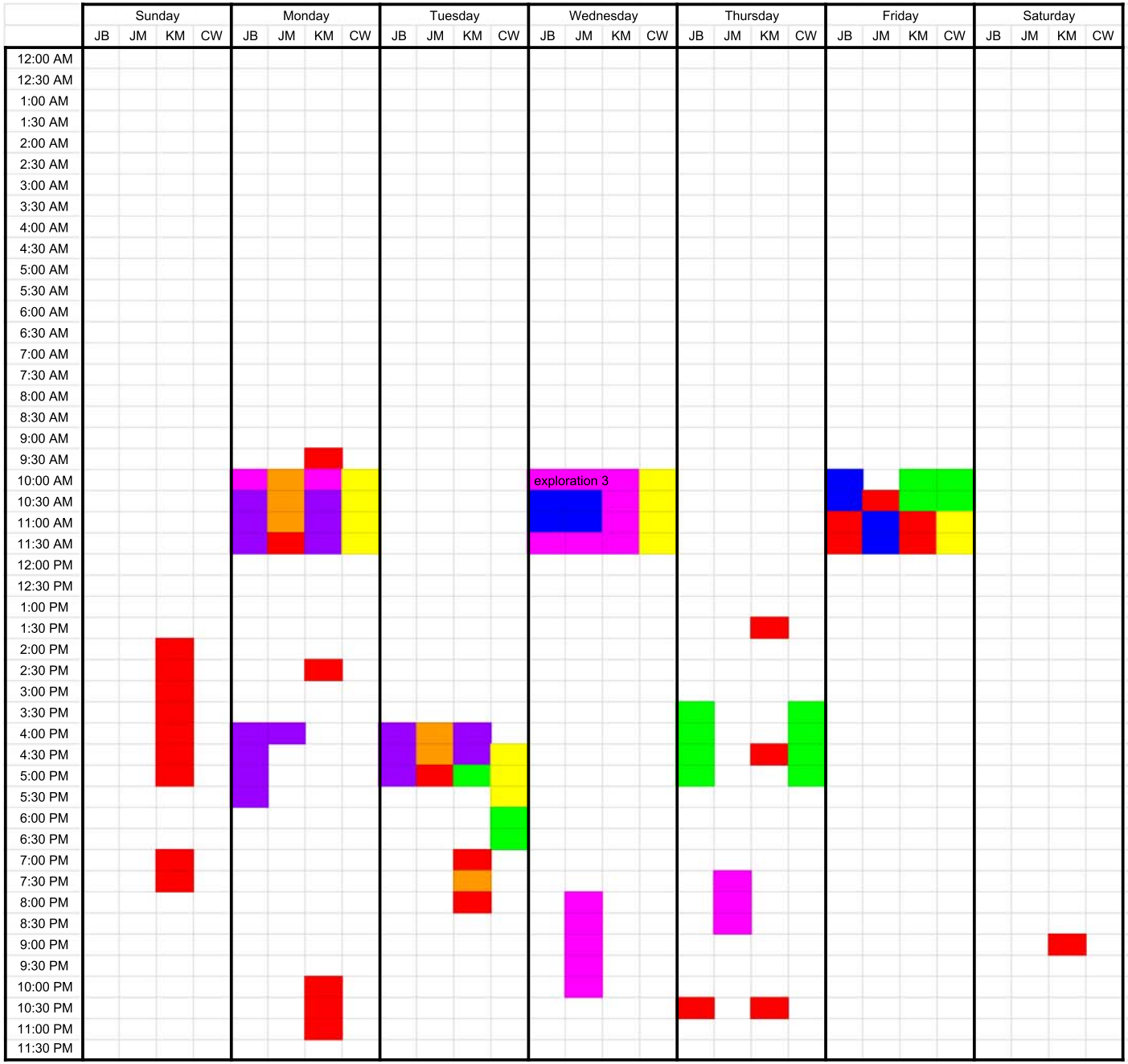
Week: Feb. 11 - Feb. 17

Category	Total	JB	JM	KM	CW
Documentation	3.5	1.5	1	1	0
Project Management	2.5	0.5	1	1	0
Coding	6.5	1	2	0	3.5
Testing	1		1	0	0
CAD	3.5		3.5	0	0
Building	1		1	0	0
Other	20.5	7	5	6	2.5
Total Hours:	38.5	10	14.5	8	6



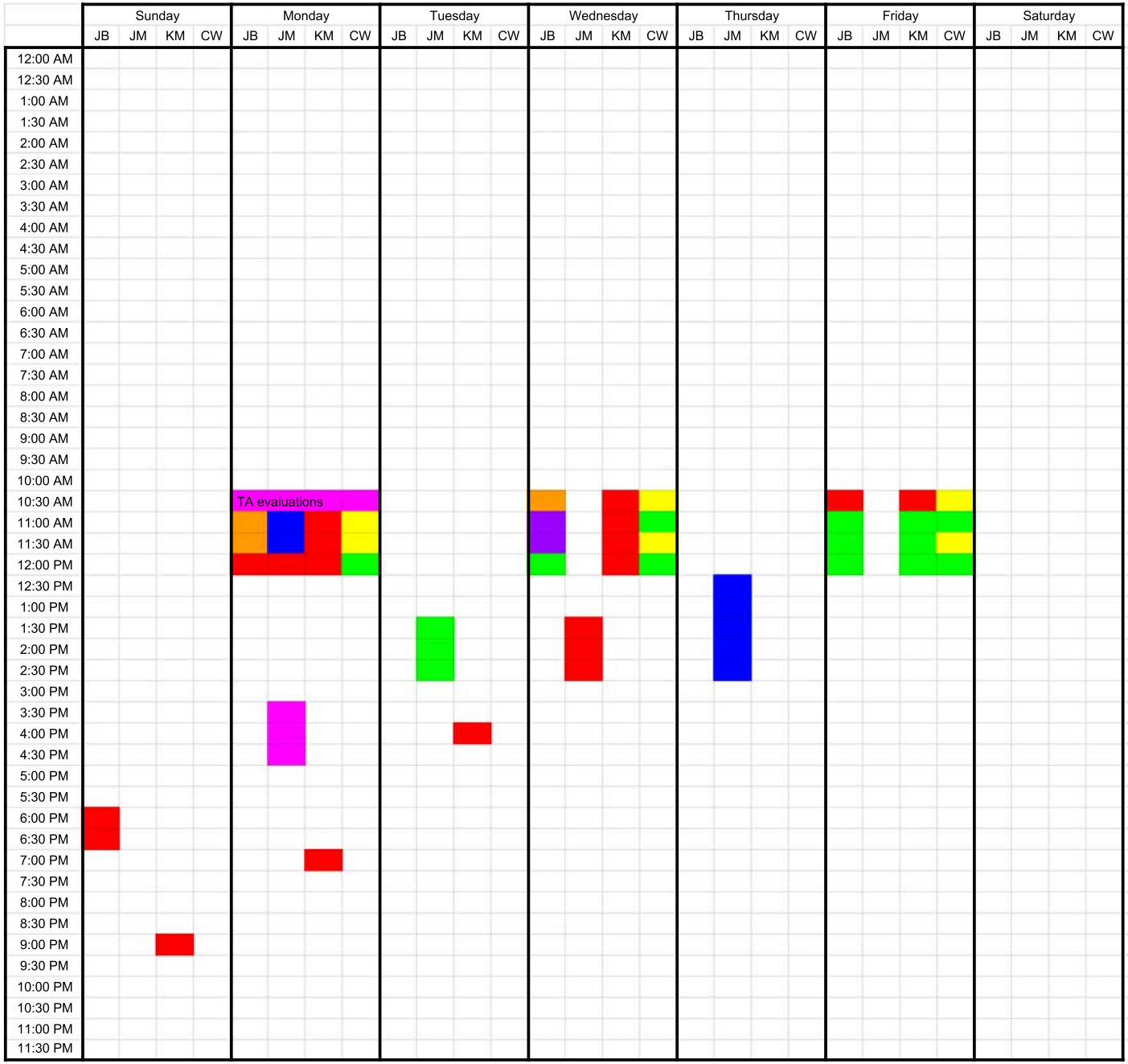
Week: Feb. 18 - Feb. 24

Category	Total	JB	JM	KM	CW
Documentation	14	3	3	8	0
Project Management	1.5	0.5	1	0	0
Coding	10	0	2.5	0	7.5
Testing	18	2	5	6	5
CAD	5	2	1	2	0
Building	12	7.5	0.5	2	2
Other	1.5	0.5	1	0	0
Total Hours:	62	15.5	14	18	14.5



Week: Feb. 25 - Mar. 3

Category	Total	JB	JM	KM	CW
Documentation	14	1.5	1.5	9.5	1.5
Project Management	1.5	0	1	0.5	0
Coding	6.5	0	0	0	6.5
Testing	9.5	2	2	1.5	4
CAD	5	2	3	0	0
Building	7.5	5	0	2.5	0
Other	7.5	1.5	3.5	2.5	0
Total Hours:	51.5	12	11	16.5	12



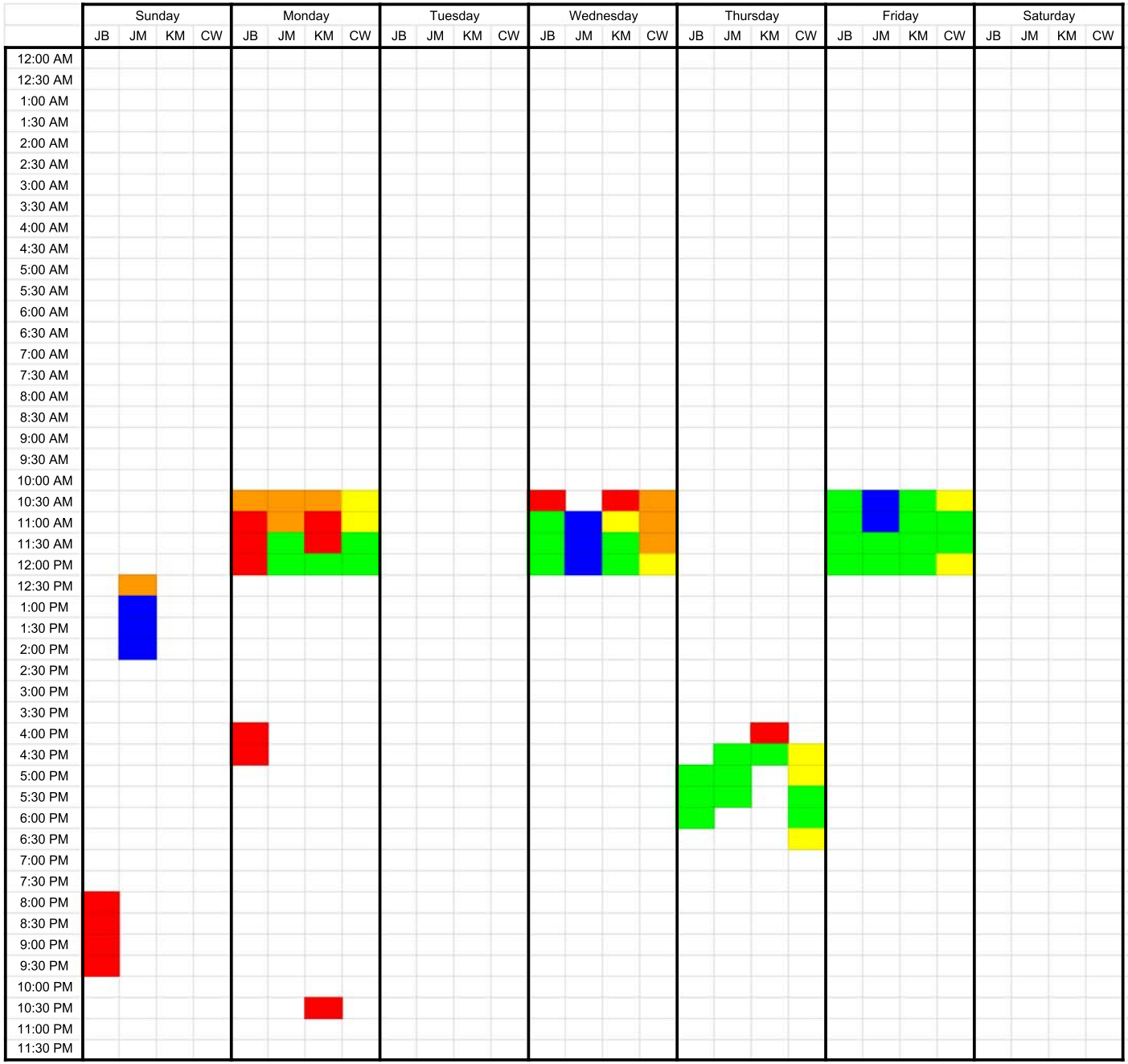
Week: Mar. 4 - Mar. 10

Category	Total	JB	JM	KM	CW
Documentation	8.5	2	1	5.5	0
Project Management	3.5	1.5	2	0	0
Coding	3	0	0	0	3
Testing	7	2	1	1.5	2.5
CAD	2	0	2	0	0
Building	1	1	0	0	0
Other	3	0.5	1.5	0.5	0.5
Total Hours:	28	7	7.5	7.5	6

	Sunday				Monday				Tuesday				Wednesday				Thursday				Friday				Saturday			
	JB	JM	KM	CW	JB	JM	KM	CW	JB	JM	KM	CW	JB	JM	KM	CW	JB	JM	KM	CW	JB	JM	KM	CW	JB	JM	KM	CW
12:00 AM																												
12:30 AM																												
1:00 AM																												
1:30 AM																												
2:00 AM																												
2:30 AM																												
3:00 AM																												
3:30 AM																												
4:00 AM																												
4:30 AM																												
5:00 AM																												
5:30 AM																												
6:00 AM																												
6:30 AM																												
7:00 AM																												
7:30 AM																												
8:00 AM																												
8:30 AM																												
9:00 AM																												
9:30 AM																												
10:00 AM																												
10:30 AM																												
11:00 AM																												
11:30 AM																												
12:00 PM																												
12:30 PM																												
1:00 PM																												
1:30 PM																												
2:00 PM																												
2:30 PM																												
3:00 PM																												
3:30 PM																												
4:00 PM																												
4:30 PM																												
5:00 PM																												
5:30 PM																												
6:00 PM																												
6:30 PM																												
7:00 PM																												
7:30 PM																												
8:00 PM																												
8:30 PM																												
9:00 PM																												
9:30 PM																												
10:00 PM																												
10:30 PM																												
11:00 PM																												
11:30 PM																												

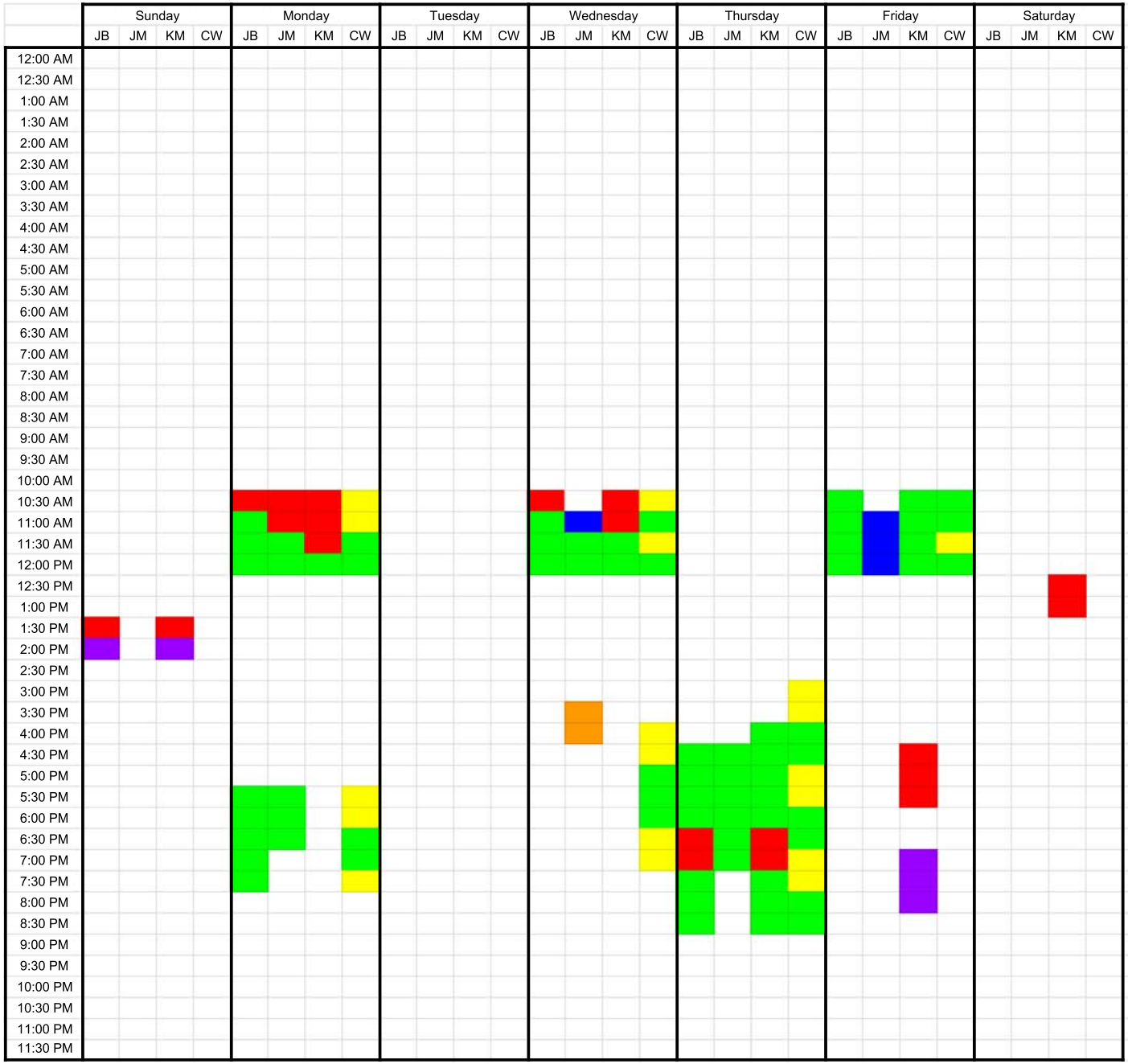
Week: Mar. 11 - Mar. 17

Category	Total	JB	JM	KM	CW
Documentation	4.5		0	4.5	
Project Management	0		0	0	
Coding	0		0	0	
Testing	0		0	0	
CAD	0		0	0	
Building	0		0	0	
Other	0		0	0	
Total Hours:	4.5	0	0	4.5	0



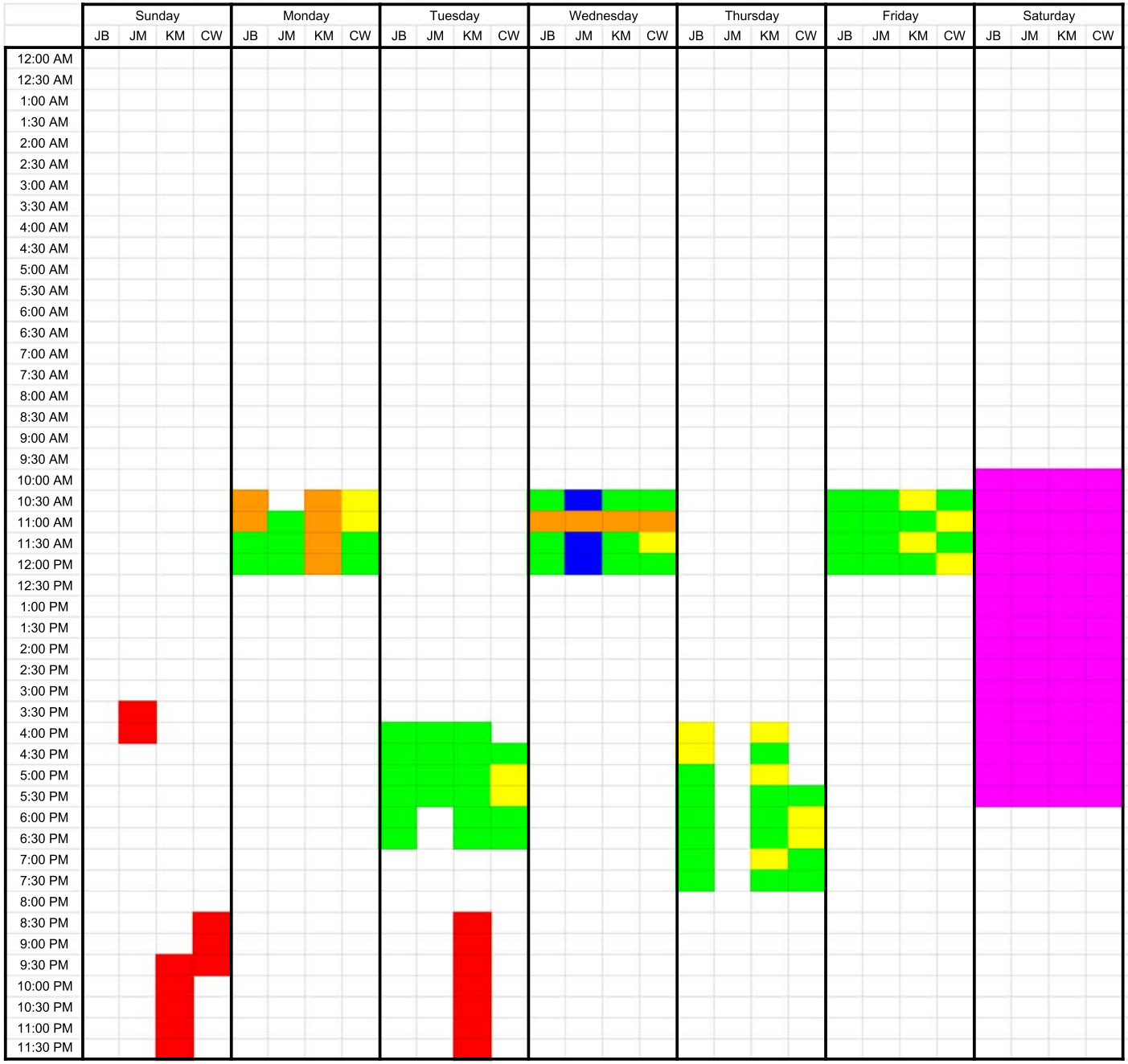
Week: Mar. 18 - Mar. 24

Category	Total	JB	JM	KM	CW
Documentation	7.5	5	0	2.5	0
Project Management	4	0.5	1.5	0.5	1.5
Coding	4.5	0	0	0.5	4
Testing	16	5.5	3.5	4	3
CAD	4	0	4	0	0
Building	0	0	0	0	0
Other	0	0	0	0	0
Total Hours:	36	11	9	7.5	8.5



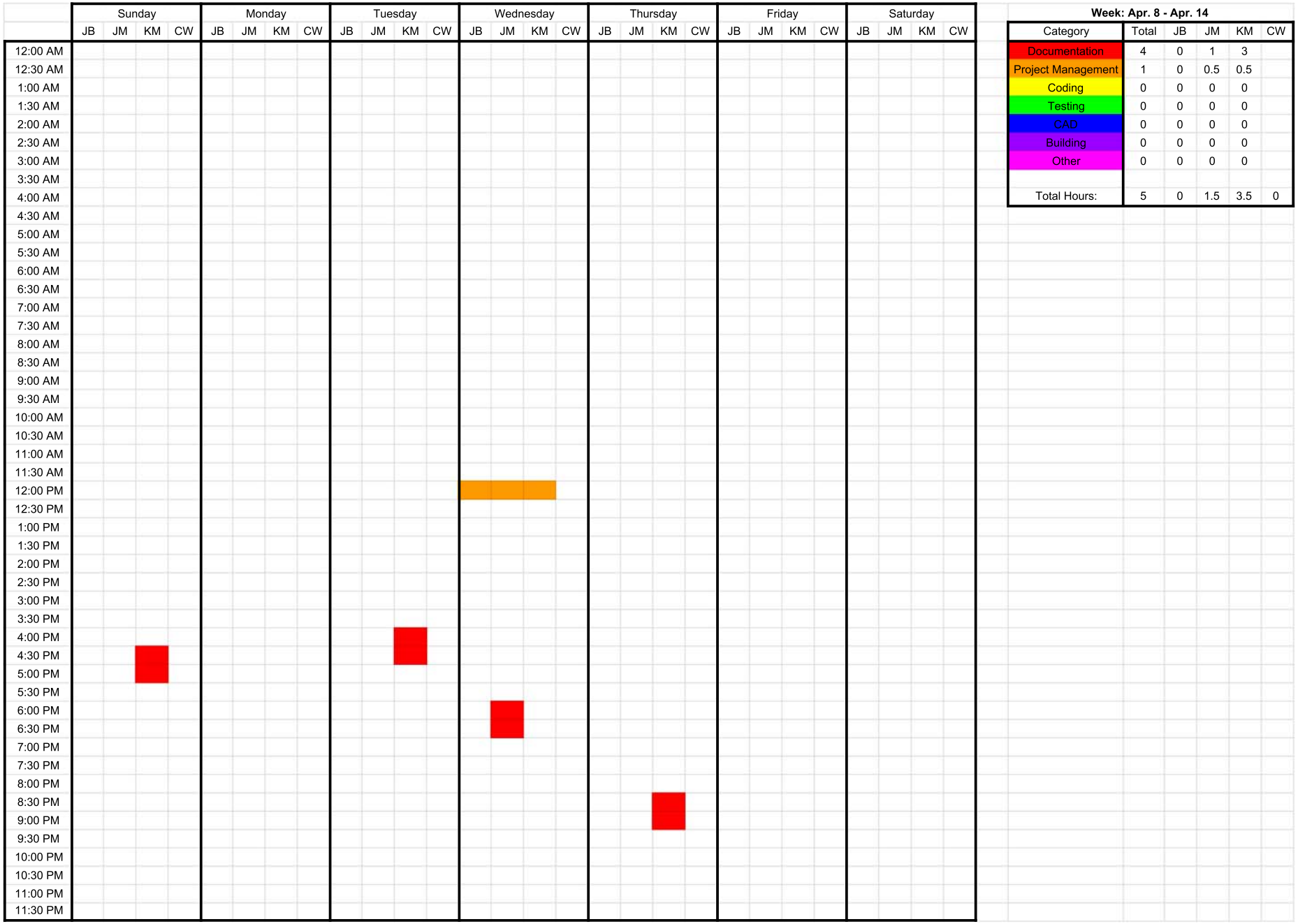
Week: Mar. 25 - Mar. 31

Category	Total	JB	JM	KM	CW
Documentation	10	2.5	1	6.5	0
Project Management	1	0	1	0	0
Coding	9	0	0	0	9
Testing	34	11	6.5	7.5	9
CAD	2	0	2	0	0
Building	2.5	0.5	0	2	0
Other	0	0	0	0	0
Total Hours:	58.5	14	10.5	16	18



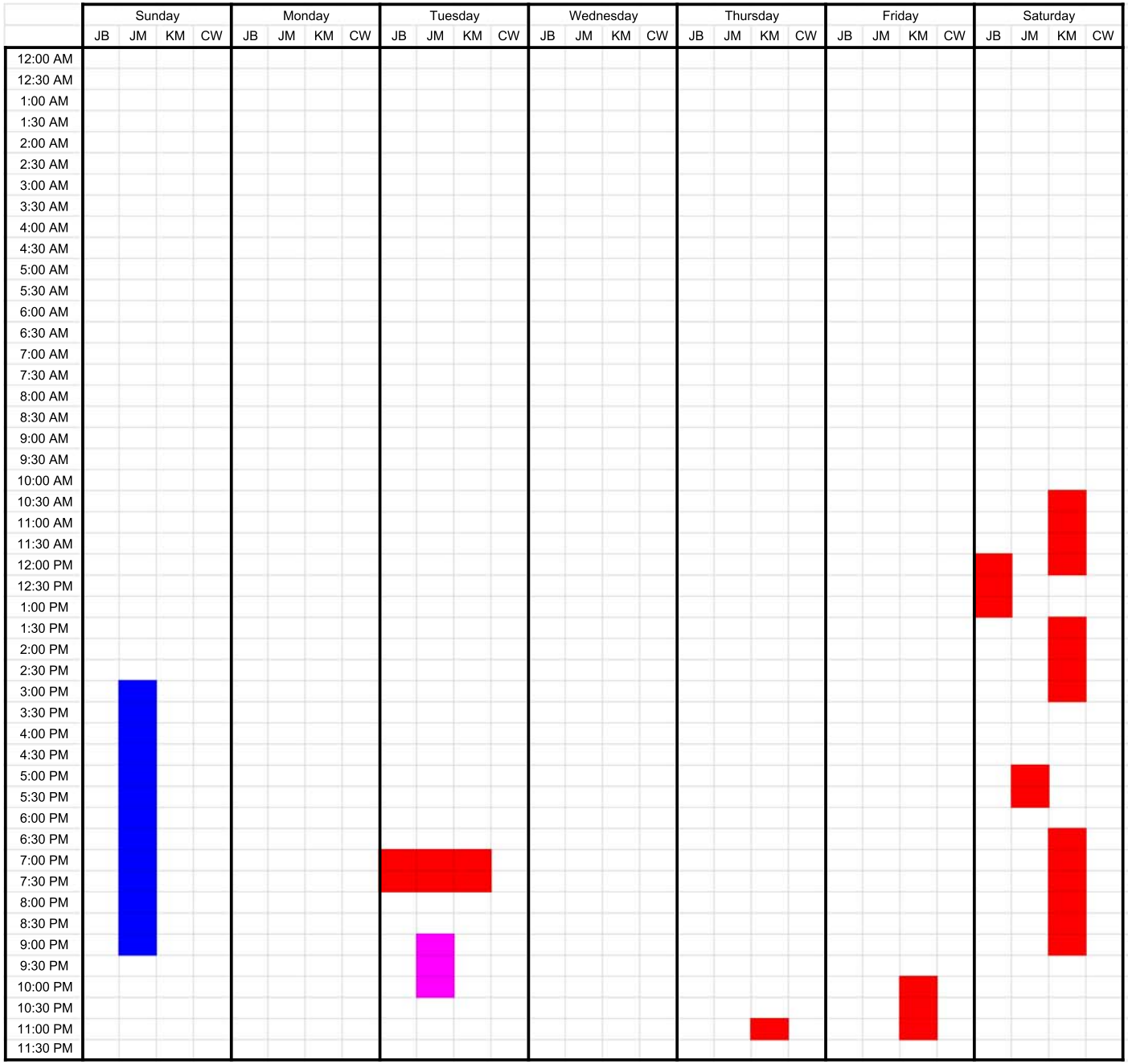
Week: Apr. 1 - Apr. 7

Category	Total	JB	JM	KM	CW
Documentation	9.5	0	1	7	1.5
Project Management	5	1.5	0.5	2.5	0.5
Coding	8	1	0	2.5	4.5
Testing	30	10.5	5.5	8	6
CAD	1.5	0	1.5	0	0
Building	0	0	0	0	0
Other	32	8	8	8	8
Total Hours:	86	21	16.5	28	20.5



Week: Apr. 8 - Apr. 14

Category	Total	JB	JM	KM	CW
Documentation	4	0	1	3	
Project Management	1	0	0.5	0.5	
Coding	0	0	0	0	
Testing	0	0	0	0	
CAD	0	0	0	0	
Building	0	0	0	0	
Other	0	0	0	0	
Total Hours:	5	0	1.5	3.5	0



Week: Apr. 15 - Apr. 21

Category	Total	JB	JM	KM	CW
Documentation	14.5	2.5	2	10	
Project Management	0	0	0	0	
Coding	0	0	0	0	
Testing	0	0	0	0	
CAD	6	0	6	0	
Building	0	0	0	0	
Other	1.5	0	1.5	0	
Total Hours:	22	2.5	9.5	10	0

	Sunday				Monday				Tuesday				Wednesday				Thursday				Friday				Saturday			
	JB	JM	KM	CW	JB	JM	KM	CW	JB	JM	KM	CW	JB	JM	KM	CW	JB	JM	KM	CW	JB	JM	KM	CW	JB	JM	KM	CW
12:00 AM																												
12:30 AM																												
1:00 AM																												
1:30 AM																												
2:00 AM																												
2:30 AM																												
3:00 AM																												
3:30 AM																												
4:00 AM																												
4:30 AM																												
5:00 AM																												
5:30 AM																												
6:00 AM																												
6:30 AM																												
7:00 AM																												
7:30 AM																												
8:00 AM																												
8:30 AM																												
9:00 AM																												
9:30 AM																												
10:00 AM																												
10:30 AM																												
11:00 AM																												
11:30 AM																												
12:00 PM																												
12:30 PM																												
1:00 PM																												
1:30 PM																												
2:00 PM																												
2:30 PM																												
3:00 PM																												
3:30 PM																												
4:00 PM																												
4:30 PM																												
5:00 PM																												
5:30 PM																												
6:00 PM																												
6:30 PM																												
7:00 PM																												
7:30 PM																												
8:00 PM																												
8:30 PM																												
9:00 PM																												
9:30 PM																												
10:00 PM																												
10:30 PM																												
11:00 PM																												
11:30 PM																												

Week: Apr. 22 - Apr. 28

Category	Total	JB	JM	KM	CW
Documentation	12.5	1.5	6	2.5	2.5
Project Management	0	0	0	0	0
Coding	0	0	0	0	0
Testing	0	0	0	0	0
CAD	0	0	0	0	0
Building	0	0	0	0	0
Other	0	0	0	0	0
Total Hours:	12.5	1.5	6	2.5	2.5

APPENDIX M

Testing Log

Date	Start Time	End Time	Tester(s)	Observer(s)	Course	What Was Tested	Why Tested	Result
2/20/2018	4:00	4:15	Colby	Kelly	F	Time-based movement	So we can move	It moves!
2/20/2018	4:20	4:35	Colby	Kelly	F	Time-based calibration to jack	Part of performance test	Makes it to jack sometimes, inconsistent
2/20/2018	4:40	4:55	Colby	Kelly	E	Time-based calibration to buttons	Part of performance test	Can reliably do buttons if the jack portion is completed
2/20/2018	5:00	5:15	Colby	Kelly	H	Movement up ramp	Test capability to move up ramp	Can inconsistently make it up ramp when power is in back
2/20/2018	5:30	5:45	Jack	Colby	H	Encoder-based movement	Looking for more consistent movement	Still slightly inconsistent
2/20/2018	6:00	6:15	Jack	Colby	H	Encoder-based calibration of course	Completing the course with more consistently	Can complete the performance test, but still very inconsistent
2/20/2018	6:30	6:45	Colby	Kelly/Jack	H	Starting on light	To ensure the CDS works	Works as desired
2/20/2018	7:30	7:45	Colby	Kelly/Jack	C	Time-based movement	See if it's more reliable than encoder	Still inconsistent
2/20/2018	8:45	9:00	Colby	Kelly	C	Skid movement	Wheels were causing inconsistencies	Drifts right, but CONSISTENT
2/22/2018	2:00	2:15	Colby	Kelly	C	Angles for performance test 1	to pass performance test 1	successful and consistent runs
2/22/2018	2:30	2:45	Colby	Kelly	C	Color reading for button pressing	to see if Dan could correctly identify the color	success on both colors
2/22/2018	2:45	3:00	Colby	Kelly	H	Power in pushing buttons for performance test 2	to pass performance test 2	success (better on blue than red, but works for both)
2/23/2018	10:30	10:40	Colby	Kelly/Jack/Justin	H	Finish calibration for performance test 2	To check consistency	Consistent, can run performance test 2
2/26/2018	10:30	10:45	Colby	-	H	Start coding main route	Not needed now, but for the future	Fast, but not fast enough
2/26/2018	10:50	11:05	Colby	-	F	Start run at an angle	Make the movement to the button board faster	Can be done, but RPS heading is desired
2/27/2018	5:00	5:15	Colby	Kelly/Jack/Justin	C	RPS testing	To see if we can use RPS	RPS is fully functional
2/27/2018	5:20	5:35	Colby	-	H	Whether we could thread and grab the wrench	We need to see if adjustments must be made	Mechanism is capable of threading, grabbing, and moving wrench
2/27/2018	5:35	5:50	Colby	-	H	Starting from start position to grab wrench	Setting up for Perfomance Test 3	Can be done, but unreliable due to servo variation. Secure better
2/27/2018	6:00	6:15	Colby	-	C	Moving up the ramp with wrench	Next portion of PT3	Robot drifts to left on ramp. Adjust motor power after fixes
2/27/2018	6:15	6:30	Colby	-	H	Grab wrench and move up ramp from start	Combining last two tests	See problems from before, but is possible. Consitency with fixes
3/1/2018	3:30	3:45	Colby	Justin	C	Adjust PT3 wrench grab for minor arm changes	Calibrate the modified arm for PT3	Much better consistency
3/1/2018	3:50	4:05	Colby	Justin	H	Adjust PT3 movement for minor arm changes	Weight differentiation, power mistake found	Functional, make sure to fully power at all times, hits button board
3/1/2018	4:10	4:25	Colby	Justin	H	New path mapping for wrench grabbing	To avoid button board	New path found, does not touch button board whatsoever
3/1/2018	4:30	4:45	Colby	Justin	C	Recalibrate bottom portion for PT3	So that the initial path is fully functional w/ new path	Wrench is now consistently grabbed
3/1/2018	4:50	5:05	Colby	Justin	C	Recalibrate upper portion for PT3	So that the full path is fully functional w/ new path	Wrench is now consistently grabbed and deposited
3/1/2018	5:15	5:30	Colby	Justin	C	Add final portion to touch fuel crank	Finish PT3	Pretty simple, works pretty easily
3/2/2018	10:25	10:40	Colby	Kelly/Jack/Justin	C	Checking path to ensure functionality on PT3	To make sure that PT3 will run without fault	Some small problems found and adjusted for, but otherwise smooth
3/5/2018	11:50	12:05	Colby	-	H	Testing start of main path, towards wrench	To start work on the main path	Numbers are off, but adjustments can be made
3/9/2018	10:50	10:55	Colby	Kelly/Justin	D	test path for PT4	beginning of PT4	adjustments are needed for multiple distances and angles
3/9/2018	11:00	11:03	Colby	Kelly/Justin	C	test power needed to get up the concrete ramp	to continue PT4	runs into the tires and drifts, but consistently
3/9/2018	11:05	11:10	Colby	Kelly/Justin	H	test route up to the ramp	continue PT4	some angles are too much; power levels greatly affect performance
3/9/2018	11:15	11:20	Colby	Kelly/Justin	H	test angle and distance leading to crank	continue PT4	not consistent
3/9/2018	11:20	11:25	Colby	Kelly/Justin	G	test movement to get down the ramp	continue PT4	need to change some distances, little consistency
3/9/2018	11:27	11:35	Colby	Kelly/Justin	A	test it all together	continue PT4	not consistent, using the grass ramp would be better
3/9/2018	11:45	11:55	Colby	Kelly/Justin	B	test new route going up the grass ramp	modify PT4	this will be better, but hits the wrench, we should use PT3 route
3/9/2018	12:20	12:25	Colby	Kelly/Justin	H	test the modified PT3 route to get up the ramp	continue PT4	good until driving toward crank, the dip in the course causes trouble
3/19/2018	10:30	10:40	Colby	Kelly/Justin	H	test the modified PT3 route to get up the ramp	continue PT4	tilts while going up the ramp
3/19/2018	11:00	11:10	Colby	Kelly/Justin	H	test power up ramp	continue PT4	power shift improved veering
3/19/2018	11:30	11:40	Colby	Jack	H	test angle to crank	continue PT4	veers while driving towards crank
3/19/2018	12:00	12:15	Colby	Jack/Justin/Kelly	H	distance to crank	continue PT4	drives correct distance
3/21/2018	10:50	11:05	Kelly	Justin	H	path down the ramp	continue PT4	bumps wall somewhat on way down
3/21/2018	11:10	11:30	Kelly	Justin	G	angle to grass ramp	continue PT4	changed angle improves line to crank
3/21/2018	11:50	12:00	Colby	Jack/Justin/Kelly	H	drifting to fuel crank	continue PT4	wheel seems to be misaligned
3/21/2018	12:05	12:10	Colby	Jack/Justin/Kelly	D	fuel crank turn	continue PT4	prongs only work when perfectly aligned
3/21/2018	12:15	12:22	Colby	Jack/Justin/Kelly	H	fuel crank turn	continue PT4	need to remove prongs and add friction surface
3/22/2018	4:20	4:26	Colby	Jack/Kelly	G	Alignment of robot	continue PT4	decided to switch out tilted wheel
3/22/2018	4:35	4:44	Colby	Jack/Kelly	G	Check new wheel	continue PT4	drives much straighter, correct issue
3/22/2018	4:50	5:00	Colby	Jack/Kelly	G	Practice PT4 run	continue PT4	ran correctly
3/22/2018	5:05	5:10	Colby	Jack/Kelly	D	Practice PT4 run	continue PT4	overturned crank
3/22/2018	5:25	5:32	Colby	Jack/Kelly	D	Practice PT4 run	continue PT4	ran correctly
3/22/2018	5:40	5:45	Colby	Jack/Justin	G	RPS testing	prep for Final run	data gathered
3/22/2018	5:55	6:00	Colby	Jack/Justin	G	RPS testing	prep for Final run	data gathered
3/22/2018	6:10	6:20	Colby	Jack/Justin	D	RPS testing	prep for Final run	successful test

Date	Start Time	End Time	Tester(s)	Observer(s)	Course	What Was Tested	Why Tested	Result
3/22/2018	6:30	6:40	Colby	Jack/Justin	G	RPS testing	prep for Final run	successful test
3/23/2018	10:40	10:50	Colby	Kelly	H	PT4 practice	PT4	not squaring up with wall
3/23/2018	11:00	11:10	Colby	Kelly	D	angle to the crank	PT4	spinning works when aligned, but not well when its not perfectly aligned
3/23/2018	11:15	11:30	Colby	Kelly/Jack/Justin	G	angle of crank turn	PT4	overspinning the crank; the back and forward doesnt seem necessary
3/23/2018	11:30	11:37	Colby	Kelly/Jack/Justin	E	angle of turn; path down ramp	PT4	overspin or underspin (whichever isnt wanted); correct path
3/23/2018	11:45	12:00	Colby	Kelly/Jack/Justin	H	crank turn	PT4	only works when perfectly aligned
3/23/2018	12:00	12:10	Colby	Kelly/Jack/Justin	H	consistency	PT4	not at all consistent, but he miraculously got it officially
3/25/2018	1:50	2:10	Justin	Kelly	G	cardbaord jig for alignment	to improve consistency	the jig does not help
3/26/2018	10:45	11:00	Colby	-	F	path to wrench	indiv comp	adjustments were made to correctly pick up the wrench using RPS
3/26/2018	11:10	11:25	Colby	-	G	path to car jack	indiv comp	too much precision leads to less accuracy
3/26/2018	11:30	11:45	Colby	-	H	lower level path	ind comp	lower level path works, but more precision on the headings
3/26/2018	11:50	12:10	Colby	-	F	lower level path	ind comp	too much precision, robot never stops
3/26/2018	12:15	12:25	Colby	Justin/Kelly	F	lower level path	ind comp	good precision and anccuracy on lower level
3/26/2018	6:00	6:10	Colby	Justin	H	lower level path and fuel crank	indiv comp	completed lower level correctly
3/26/2018	6:15	6:25	Colby	Justin	H	lower level path and fuel crank	indiv comp	completed lower level correctly except for wrench
3/26/2018	6:30	6:40	Colby	Justin	E	lower level path and fuel crank	indiv comp	completed lower level correctly
3/26/2018	6:45	6:55	Colby	Justin	G	lower level path and fuel crank	indiv comp	completed lower level correctly except for wrench
3/26/2018	7:00	7:10	Colby	Justin	C	lower level path and fuel crank	indiv comp	completed lower level correctly except for wrench
3/26/2018	7:10	7:20	Colby	Justin	C	lower level path and fuel crank	indiv comp	completed lower level correctly except for wrench
3/26/2018	7:20	7:30	Colby	Justin	H	lower level path and fuel crank	indiv comp	completed lower level correctly except for wrench
3/26/2018	7:30	7:40	Colby	Justin	C	lower level path and fuel crank	indiv comp	completed lower level correctly except for wrench
3/28/2018	11:15	11:20	Colby	Justin/Kelly	G	main code up to buttons	individual comp	need to adjust numbers especially on upper level
3/28/2018	11:25	11:35	Colby	Justin/Kelly	F	main code up to buttons	ind comp	too much back up after fuel crank, not orienting to button light
3/28/2018	11:43	11:50	Colby	Justin/Kelly	E	main code including buttons	ind comp	upper level not consistent, wrench back up never reached, button code needs adjusted
3/28/2018	12:00	12:05	Colby	Justin/Kelly	E	main code, buttons first	ind comp	never aligns with buttons
3/28/2018	12:10	12:15	Colby	Justin/Kelly	F	button alignment	ind comp	failed to start with light, failed to press white button
3/28/2018	12:17	12:19	Colby	Justin/Kelly/Jack	H	button alignment	ind comp	takes too long to line up
3/28/2018	12:20	12:25	Colby	Justin/Kelly/Jack	B	button alignment	ind	takes too long to line up, too much correction
3/29/2018	3:00	3:15	Colby		H	button alignment	ind	add sleep to increase speed, alter angles to line up based on color
3/29/2018	3:20	3:30	Colby		F	button alignment	ind	buttons good!
3/29/2018	3:45	3:50	Colby		G	lower level	ind	work on angles and distances to wrench
3/29/2018	3:55	4:00	Colby		B	lower level	ind	work on angles on lower level
3/29/2018	4:05	4:20	Colby	Jack/Kelly	H	lower level	ind	good with buttons (but slow), wrench and ramp distance/angle need work
3/29/2018	4:27	4:32	Colby	kelly	H	lower level up ramp	ind	need to add length before button light, add distance to wrench, too slow up ramp
3/29/2018	4:35	4:45	colby	jack	G	lower level up ramp	ind	not pressing white button!!!
3/29/2018	5:00	5:15	Colby	justin	H	lower level up ramp	ind	still not pressing white button, go in at an angle
3/29/2018	5:25	5:40	Colby	justin	F	lower level up ramp	ind	white button troubles remain, angle to wrench not perfect
3/29/2018	5:45	5:50	Colby	kelly	H	full course	ind	white button trouble, wrench angle, gets caught on upper level grass
3/29/2018	5:55	6:00	Colby	justin	H	full course	ind	white button (needs more power), wrench RPS coordinate, crank angle
3/29/2018	6:00	6:03	Colby	justin	G	white button, wrench	ind	turn to white button needs to be greater, drive more towards jack before turning to wrench,
3/29/2018	6:06	6:10	colby	justin	D	white button, wrench, and ramp	ind	wrong way to buttons...failed run
3/29/2018	6:15	6:20	colby	justin	H	buttons, wrench, and ramp	ind comp	needs more back up before button, still short on wrench
3/29/2018	6:23	6:30	colby	justin	G	buttons and wrench	ind comp	buttons good! still not getting wrench, crank heading, final drive too much
3/29/2018	6:35	6:50	colby	justin	E	wrench and crank RPS	ind comp	white button good! too close to the wrench, needs to drive further into garage
3/29/2018	6:55	7:00	colby	justin	G	wrench and crank RPS	ind comp	too far on the wrench, almost got fuel crank (not enough spin)
3/29/2018	7:06	7:10	colby	kelly	E	wrench and crank RPS	ind comp	failed heading checks; need to undo the counts on heading checks that we added
3/29/2018	7:15	7:20	colby	kelly	H	full course	ind comp	failed white button, too far on wrench, swerved to crank, wheel fell off
3/29/2018	7:30	7:35	colby	kelly	H	full course	ind comp	failed white button, wrench servo too low, RPS x/y wasn't working
3/29/2018	7:43	7:51	colby	justin	H	full course w/ RPS x/y working	ind comp	failed white button, wrench servo too low
3/29/2018	8:00	8:09	colby	kelly/justin	G	full course	ind comp	failed white button, failed wrench alignment, failed crank alignment
3/29/2018	8:15	8:20	justin	kelly	H	full course	ind comp	got white button but wrench not aligned and servo too high
3/29/2018	8:25	8:30	colby	justin/kelly	H	full course	ind comp	didnt get buttons because of added wood pieces, not RPS on top failure

Date	Start Time	End Time	Tester(s)	Observer(s)	Course	What Was Tested	Why Tested	Result
3/29/2018	8:40	8:45	Colby	justin/kelly	H	full course	ind comp	failed to get white button, whole route was messed up (because of charge?)
3/29/2018	8:53	9:00	colby	justin/kelly	H	full course	ind comp	same problems as before...
4/2/2018	11:00	11:10	Colby	Justin	H	full course	final competition	discovery that QR code was placed back on backwards
4/2/2018	11:10	11:20	Colby	Justin	H	full course	final competition	got buttons, wrench, jack, missed crank
4/2/2018	11:20	11:30	Colby	Justin	G	full course	final competition	got buttons, wrench, jack, missed crank
4/2/2018	11:30	11:40	Colby	Justin/Jack	G	full course	final competition	got buttons, wrench, jack, missed crank
4/2/2018	11:40	11:50	Colby	Justin	C	full course	final competition	got buttons,jack, missed crank, jack, stuck on hill
4/2/2018	11:50	12:00	Colby	Justin	G	full course	final competition	got buttons, wrench, jack, missed crank, stuck on hill
4/2/2018	12:00	12:10	Colby	Justin	H	full course	final competition	got buttons, wrench, jack, missed crank
4/2/2018	12:10	12:20	Colby	Justin	H	full course	final competition	got buttons, wrench, jack, missed crank, did not make it to red button
4/2/2018	12:20	12:25	Colby	Justin	H	full course	final competition	got buttons, wrench, jack, missed crank
4/2/2018	6:40	6:50	Colby	Justin	H	full course	final competition	need to adjust distances and turns after replacing motor hubs
4/2/2018	6:50	7:00	Colby	Justin	H	full course	final competition	need to adjust distances and turns after replacing motor hubs
4/2/2018	7:00	7:10	Colby	Justin	H	full course	final competition	need to adjust distances and turns after replacing motor hubs
4/2/2018	7:10	7:20	Colby	Justin	C	full course	final competition	got buttons, wrench, issues on ramp
4/2/2018	7:20	7:30	Colby	Justin	E	full course	final competition	got everything except crank
4/2/2018	7:30	7:40	Colby	Justin	H	full course	final competition	got everything except crank
4/2/2018	7:40	7:50	Colby	Justin	E	full course	final competition	got everything except crank
4/3/2018	4:33	4:36	Colby	Jack/Kelly	F	full course	final competition	went up ramp wildly, didn't get crank, didn't make it to final button (inconsistent turns)
4/3/2018	4:40	4:46	Colby	Kelly	B	full course	final competition	went up ramp wildly, didn't get crank, didn't get RPS button (increase distance to blue button)
4/3/2018	4:52	4:56	Colby	Kelly	B	full course	final competition	nice up ramp, aligned with crank but did not turn all the way, wild down ramp
4/3/2018	5:10	5:15	Colby	Justin/Kelly	G	full course	final competition	nice on lower level, did not align with crank or garage, bad turns to final button
4/3/2018	5:30	5:37	Colby	Justin/Kelly	C	full course	final competition	didnt get wrench (add distance), didnt align with ramp (adjust RPS)
4/3/2018	5:40	5:45	Colby	Kelly	H	full course	final competition	didnt get wrench, aligned with ramp on double RPS check, didnt align with garage
4/3/2018	5:53	5:58	Colby	Kelly/Justin	E	full course	final competition	didnt get buttons or wrench, didnt align with crank with or without RPS
4/3/2018	6:05	6:11	Colby	Kelly/Justin	G	full course	final competition	didnt get wrench, didnt go far enough up ramp to align with crank, too much turn into garage
4/3/2018	6:14	6:17	Colby	Kelly/Justin	H	full course	final competition	didnt get RPS button or wrench and upper level was a mess
4/3/2018	6:24	6:30	Colby	Kelly/Justin	F	full course	final competition	didnt get wrench, bad turn to ramp
4/3/2018	6:35	6:40	Colby	Kelly/Justin	F	full course	final competition	didnt get wrench or crank or anything
4/3/2018	6:50	6:55	Kelly	Justin	D	ind comp code	final competition	got everything except the wrench
4/3/2018	7:00	7:05	Justin	Kelly	H	slight change on wrench	final competition	got everything except fuel crank
4/4/2018	11:15	11:20	Kelly	Justin	F	full course	final competition	remove long RPS check
4/4/2018	11:22	11:27	Kelly	Justin	C	full course	final competition	add heading check for garage and eliminate swerves
4/4/2018	11:55	11:57	Justin	Kelly	E	full course	final competition	eliminate start up power
4/4/2018	12:02	12:07	Justin	Kelly	C	full course	final competition	heading check by garage, swerves by control panel
4/4/2018	12:12	12:18	Kelly	Justin	F	full course	final competition	adjust non-RPS code and garage heading
4/5/2018	4:08	4:12	Kelly	-	H	full course	final competition	adjust non-RPS code, wrench RPS coordinates, add more power on buttons
4/5/2018	4:21	4:25	Kelly	-	F	full course	final competition	adjust garage, check the blue button values
4/5/2018	4:30	4:40	Kelly	-	G/E	full course	final competition	wrench RPS coordinate off, top with RPS is off
4/5/2018	4:50	4:53	Kelly	-	E	full course	final competition	doesn't read blue button or align with wrench
4/5/2018	5:05	5:10	Kelly	Justin	C/D	full course	final competition	go further up the ramp with RPS or turn less without RPS
4/5/2018	5:25	5:30	Justin	Kelly	H	full course	final competition	blue reading is bad, non-RPS garage turn too much
4/5/2018	5:38	5:50	Justin	Kelly	E/G	full course	final competition	back up less from crank, non-RPS crank line up, swerved at buttons
4/5/2018	5:58	6:05	Colby	Justin/Kelly	H	wrench correction	final competition	non-RPS turn to crank needs to be less but wrench correction works!
4/5/2018	6:15	6:18	Colby	Justin/Kelly	D	full course	final competition	further to button and jack
4/5/2018	6:25	6:40	Colby	Justin/Kelly	C/A	full course	final competition	missed garage right and missed crank right
4/5/2018	6:50	6:57	Colby	Justin/Kelly	D	full course	final competition	good, just fuel turn not 180
4/5/2018	7:00	7:07	Colby	Justin/Kelly	G	full course	final competition	code mess up caused it to get stuck
4/5/2018	7:15	7:20	Colby	Justin/Kelly	D	crank line up correction	final competition	tried to destroy the crank with the correction
4/5/2018	7:30	7:40	Colby	Justin/Kelly	G/B	full course	final competition	can't make it up the hill
4/5/2018	7:45	7:50	Colby	Justin/Kelly	G	full course	final competition	swerves on hill, wrong button color
4/6/2018	10:27	10:40	Kelly	Justin	C/D	full course	final competition	can't go up the ramp, consider RPS check at crank

Date	Start Time	End Time	Tester(s)	Observer(s)	Course	What Was Tested	Why Tested	Result
4/6/2018	10:50	11:00	Justin	Kelly/Jack	B/G	full course	final competition	can't go up ramp, RPS button not being pressed with blue
4/6/2018	11:07	11:12	Justin	Kelly/Jack	F	ramp correction	final competition	made it up the ramp, not getting blue button
4/6/2018	11:17	11:23	Colby	Justin/Kelly/Jack	H	full course	final competition	increase non-RPS angle and RPS distance up ramp
4/6/2018	11:33	11:39	Colby	Justin/Kelly/Jack	G	full course	final competition	stuck on ramp
4/6/2018	11:46	11:55	Colby	Justin/Kelly/Jack	G	full course	final competition	the top values are all off
4/6/2018	12:00	12:07	Colby	Justin/Kelly/Jack	H	full course	final competition	the top values are still off due to correction
4/6/2018	12:13	12:21	Colby	Justin/Kelly/Jack	G	full course	final competition	non-RPS values off, caught on the wall, garage turn is off