Design Phase One

Chase Garner, Sawyer Hall, Lexi Sheeler, Daniel Summers
Electrical and Computer Engineering Department
Tennessee Technological University
Cookeville, TN
ncgarner42@tntech.edu, sjhall43@tntech.edu, aasheeler42@tntech.edu, dbsummers42@tntech.edu

I. Introduction

A. Problem Definition

The South Eastern Conference (SECon) is a yearly IEEE event comprised of several competitions. One such competition is the hardware competition, where students from many schools across the region design and build a robot to compete in a unique challenge. Because the conference is being held in Mobile, Alabama, the competition is Mardi Gras themed to reflect the history and origin of the festival. This year's hardware competition requires the robot to navigate an L-shaped street within a three-minute time limit. In addition to navigation, the robot can complete challenges to earn points. The tasks include: removing beads from a "tree", placing the beads in "trash bins" or shooting them into fishnets, and moving a marshmallow off the roadway into gaps in a barricade.

There are several rules and specifications that the design must adhere to. The first and potentially most important constraint to meet is that the robot requires complete autonomy. To meet this constraint, the robot must be capable of decision-making to identify randomly placed obstacles and challenges. In addition, the robot must be contained within one cubic foot at the start of the competition, but may expand beyond this specification after the run has started. It is also specified that the robot may not break apart and must remain one unit for the entirety of the run. Finally, the robot may not cause damage to the track or harm bystanders. Any violation of these constraints will result in immediate disqualification at the judges' discretion.

B. Conceptual Design

At the beginning of the design process, key competition rules and point gaining tasks were identified. The point breakdown played into the first design phase in several ways. The scoring scheme was broken down and the maximum number of achievable points was determined. This allowed for selecting which way the robot could earn maximal points without the need to complete every task outlined in the guidelines. This led to a design that is targeted towards shooting beads into the nets instead of trying to place them into the cups along the sidewalk area. Along with this decision, the team also elected to not place high importance on removing the marshmallow from the track because it is only a one-point deduction if left on the track. No attainable designs were identified by the team, and focus was instead placed on areas which are integral to being competitive, such as the navigation system and drive train.

The robot is broken down into seven systems: main controller, power, navigation, drive train, appendages, shooting mechanism, and peripherals. Each system is then broken down further into atomic subsystems. There will be three microcontrollers: the main controller, a navigation controller, and a peripheral controller. Serial communication will be used between the main and navigation controllers, and the peripheral controller will only require power to operate. The main controller is responsible for interconnects between systems, excluding power, which will directly connect with all systems. For the next phase of the design process, component specifications will be defined first at an atomic level, then working up until the whole system has been defined.

C. Broader Impacts

Beyond the competition itself, this project aims to serve as a way to demonstrate the quality of both the Tennessee Tech Electrical and Computer Engineering (ECE) Department and the Tennessee Tech IEEE. The Tennessee Tech IEEE chapter has a longstanding tradition of fielding competitive, winning teams at SECon, and this project serves to continue such a tradition. Additionally, due to the current pandemic, interest and involvement in the IEEE chapter has massively decreased over the last two years. This project will generate more interest in IEEE from ECE students while also providing IEEE members with hands-on experience with robotics and programming. Because of the involvement of the IEEE chapter, the design of the peripheral system was changed to use its own microcontroller and an analog circuit to control the LEDs. This allows the IEEE members to work on these systems independently of the robot's development, and it ensures they receive experience in elements of both electrical and computer engineering by incorporating both analog and digital circuitry and programming.

D. Larger Process

The Design Phase I document is written with the intent of removing ambiguity and presenting a more clearly defined system. The document should leave the reader with a clearer understanding of how the competition has led the team to make the design choices presented in the following sections.

In addition, ideas presented in this document will be further built upon in future design phases.

II. Chassis

The robot's chassis will be divided into a lower section, which will house the majority of the electronics, and an upper section, which will serve as a platform for the robotic arm and shooting mechanism. The chassis and all external systems must conform to the 1 cubic foot constraint set by the competition. Because of its size and simple structure, it is intended for the chassis to be constructed with acrylic sheets instead of 3D printing it, but additional material options are still being considered. The main justification for this decision is the team's desire to avoid possible 3D printer errors that are likely to occur during the longer print times that would be necessary for the larger aspects of the chassis. In order to verify that the chassis conforms to the 1 cubic foot limit while still maintaining enough volume to house the robot's internal components, the chassis will need to be a somewhat detailed 3D model. Additionally, the size of the robot should be minimized in order to allow for a smaller turning radius.

The lower chassis (Figure 1) will be a stationary platform in order to minimize the amount of turning required. With a 1 square foot footprint, the robot would be over 16 inches diagonally. This could prevent the robot from turning within the 16-inch width of the track. Instead, the lower chassis will only be required to turn during the two 90 degree turns in the track, where the effective width is much wider. The lower chassis will contain the drivetrain, navigation sensors, each of the microcontrollers, and the power supply circuitry, including the battery. The main purpose of the lower chassis will be to house these components, but it will also be responsible for collecting the marshmallow obstacle from the track. It is the intention of the team that collecting and pushing the marshmallow will be handled by a passive system that will be incorporated in the chassis' shape and design, but the exact specifications of such a passive system must still be determined. The main constraint of the lower chassis will be the size limitation of the competition. The chassis, including the wheels on each side and the passive marshmallow system on the front, must all fit within the 1 square foot footprint of the robot.

Instead of requiring the entire robot to turn and face each target, the upper half of the chassis will be constructed in such a way that it will freely rotate at least 180 degrees to allow for the shooting mechanism to "aim" in each direction. However, the rotation must not interfere with the wiring connections between the lower and upper chassis. This turntable will house the robotic arm and the shooting apparatus. Additionally, the upper chassis (Figure 3) will serve as a storage mechanism for the beads between being collected and being shot into the nets (Figure 2). Upon being placed into the storage mechanism, the beads must be sorted into three separate containers to be individually loaded into the shooting apparatus. The number of containers was chosen in order to comply with the competition rules stating that additional points would be awarded for each

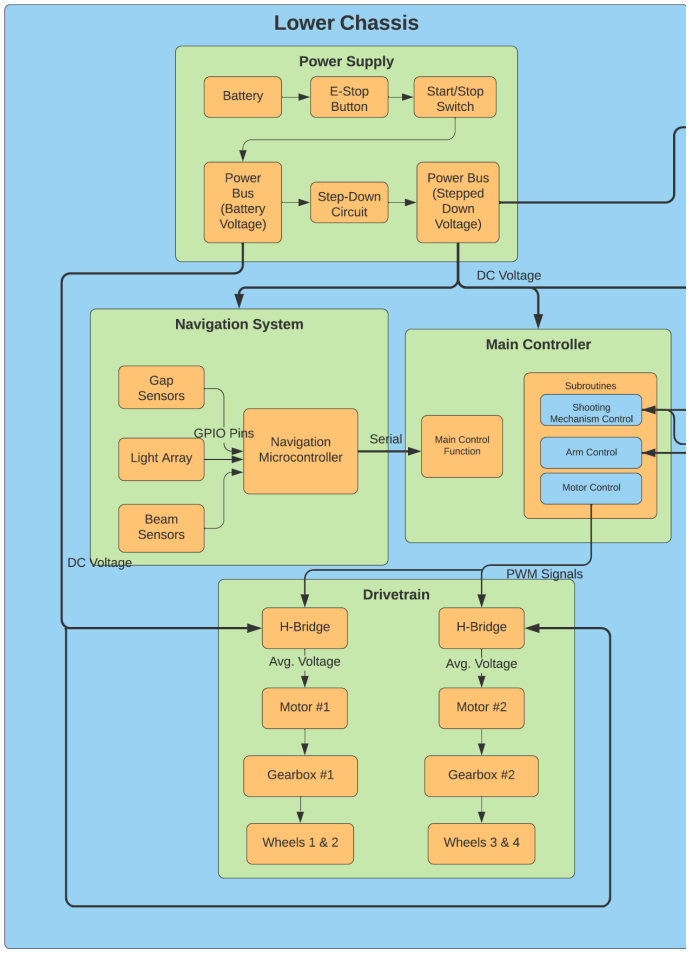

Fig. 1. Lower chassis block diagram

net containing at least one bead. Because there are three nets, three was chosen for the number of containers. These containers will be bottomless, but the housing of the shooting mechanism will serve as a temporary bottom. To "reload", the containers will be moved over an opening in the housing, allowing the beads to fall onto the belt. However, this method must be tested to ensure that the beads consistently do not get stuck in the sorting mechanism. In order to test the most optimal sorting configuration, a physical mock-up of the system must be constructed. The dimensions of the divider will then be experimentally varied and the percentage of beads in each container, as well as the percentage of beads that get stuck, will be measured. The configuration that yields the most balanced percentage of sorted beads and the lowest percentage of stuck beads will be chosen for the final design.

III. Central Controller

The central controller (Figure 4) of the robot will consist of a single Arduino Mega 2560 microcontroller. This specific hardware component was chosen for its robust functionality, simple programmability via Arduino's predefined libraries, and large number (54) of General-Purpose Input-Output (GPIO) pins, of which 15 can be used to generate Pulse Width

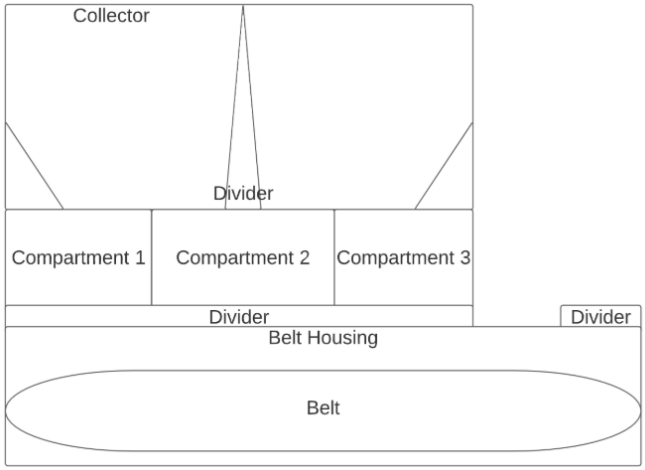

Fig. 2. Loading mechanism

Modulation (PWM) signals. The microprocessor operates at 5 Volts and is recommended to be powered by 7-12 volts which will be provided by the step-down circuitry of the robot's power supply [1]. Additionally, a serial connection must be established between the central controller and the navigation system. This serial connection will relay commands from the navigation system's microcontroller to the central controller, where they will be interpreted in order to determine the proper control subroutines to be run. This interpretation and serial communication will be handled by the central controller's main control function, which will then trigger a number of separate subroutines that are responsible for the central controller's outbound communication to the motors, arm, and shooting mechanism.

The motor subroutines will communicate with the drive-train's motor drivers via its GPIO pins to achieve the desired speed and direction of each of the two motors. This functionality will include both starting and stopping the motion of the robot, navigating the 90 degree turns in each direction, and propelling the robot both forward and backwards along the track. The arm subroutines will function by using inverse kinematics to provide the necessary positional information to each of the arm's servo motors to achieve the desired endpoint position of the arm. Such routines will include picking the beads from each tree and placing the beads in the chassis' storage container. This information will be provided to each individual servo via dedicated GPIO pins. Lastly, the subroutines related to the shooting mechanism will be responsible for rotating the upper platform of the chassis to "aim" the mechanism in the correct direction, providing a signal to the shooting belt's motor, and "reloading" the mechanism by shifting the storage container to allow for the next set of beads to fall onto the belt. Each of these motor control subroutines will drive separate PWM pins of the central controller. Consequently, each of these subroutines will accept additional feedback inputs from these external systems. This feedback will be used to align the subroutine's expected

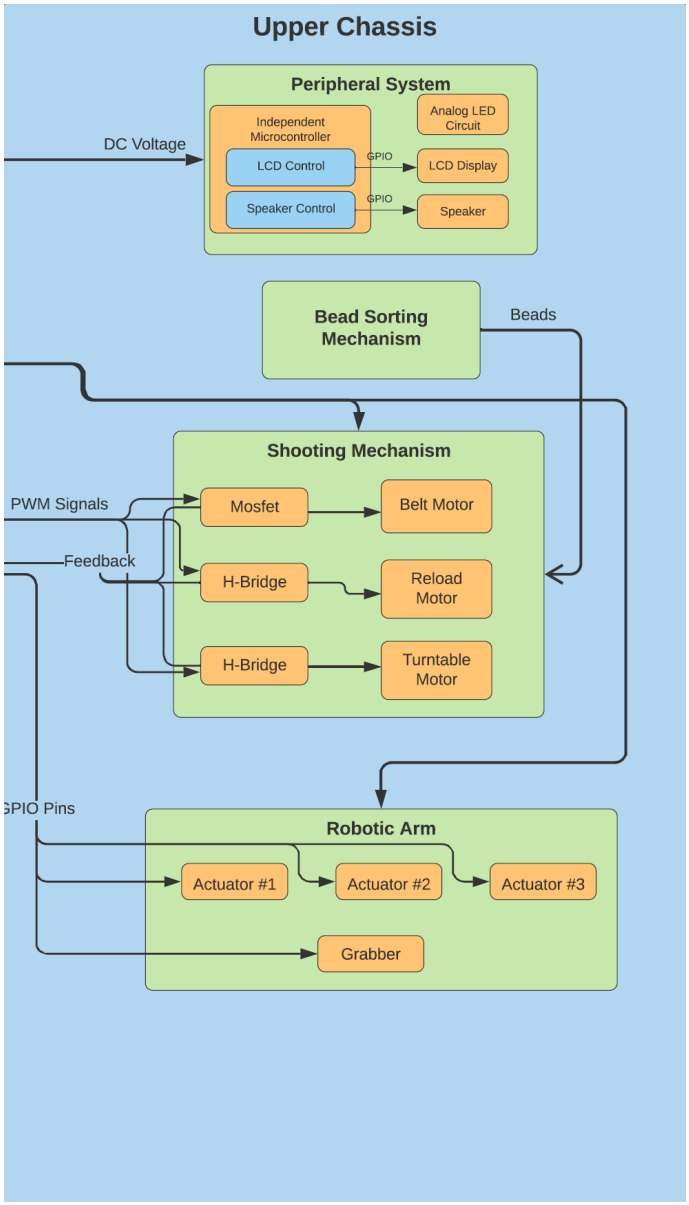

Fig. 3. Upper chassis block diagram

behavior with the actual behavior of the mechanical parts and will be provided via additional GPIO pins.

Because of the software-based nature of the central controller, the elements of the programming language can be assumed to behave in the way they are defined by the language. However, the software's behavior must be closely emulated when simulating and testing the various components driven by the software. Without closely representing the software during this testing, the components under test can not be guaranteed to behave in the same manner when they are controlled by the software. In addition, extensive experimental testing will be required to adjust the software to meet specific needs of the robot. For instance, timing tests must be conducted to ensure that enough time has passed for all of the beads to be launched by the shooting mechanism before the motors are turned off.

Lastly, the datasheet of the Arduino must be thoroughly read and understood to ensure that the microcontroller is capable of producing the output voltages and currents required by the component hardware.

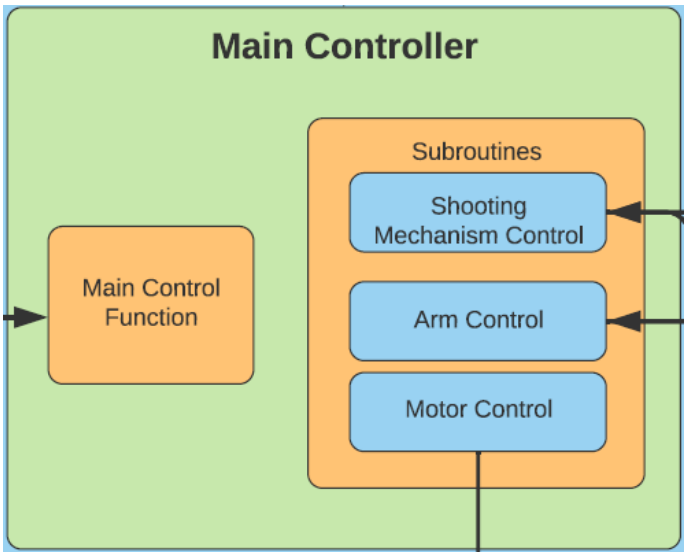

Fig. 4. Central controller block diagram

IV. Power

The battery voltage will be determined by what is needed to power the DC motors. Because of competition specifications, the robot is required to have a visible and easily accessible emergency-stop. To ensure no damage to components, the emergency stop is intended to stop only moving parts of the robot. Therefore, the emergency-stop will be the interim between the battery and motors. The emergency stop will be utilized to stop the robot in case of a major malfunction. Like the emergency stop, the competition specifications require the robot to have a clear start/stop switch. The start/stop switch will then be connected from the emergency stop to the power bus. The power bus will distribute the primary voltage to the step-down circuit and the H-bridge. The H-bridge will then connect the needed voltage to the DC motors. The step-down circuit will then decrease the battery voltage to the needed voltages for the rest of the components. The step-down circuit will be purchased and not built by the team. The step-down circuit will then be connected to two separate power buses. The peripherals will be connected to one power bus, while the navigation sensors and main controller will be connected to the other. The power circuit will need to be tested through the utilization of LTSpice. The team will test the step-down circuit and will then know that the systems work correctly by finding the output voltage and current at the step-down circuit.

V. Navigation

The navigation system (Figure 6) is responsible for identifying objects and making decisions based on inputs from a light array, beam sensors, and gap sensors. These decisions will

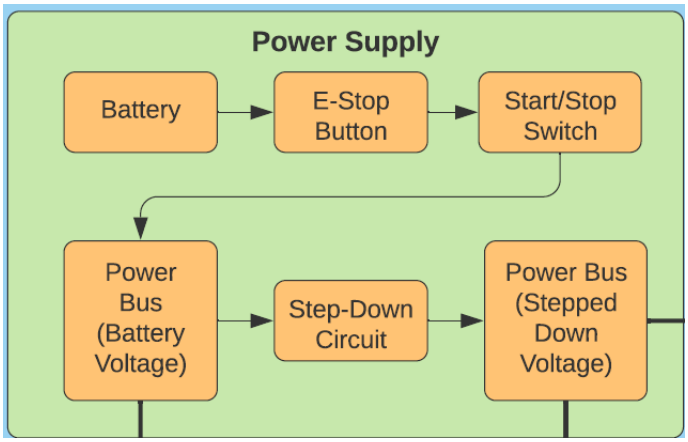

Fig. 5. Power system block diagram

then be sent to the main microcontroller via serial communication and relayed to the necessary systems. All transmissions between these microcontrollers will include an acknowledge message when the receiving controller accepts the instruction message.

Due to high computational demand and number of available ports, it was decided that the navigation system should include its own microcontroller as to not overwhelm the main controller. The navigation system will require constant monitoring of the sensors, as well as decision making based on the inputs, making this a completely separate system, except for serial communication with the main microcontroller.

The navigation microcontroller will enable interrupts from sensors after the turn around at the end of the track. If the sensor detects specific objects, it will interrupt the drive commands and prompt the correct subroutine. This information will be sent to the main microcontroller in a series of commands: adjust left, adjust right, forward, backward, left turn (90°), right turn (90°), ready to shoot, and arm ready for pick-up/drop. These commands will be appropriately carried out by the main microcontroller and other necessary systems.

The light array will be mounted on the underside of the base to assist with track traversal, using the painted white line on the track. The array will allow for small deviation from the painted white line, but will trigger left and right adjustment commands to be sent to the main microcontroller if it strays too much. If the sensor detects a 90° angle in the line, it will send a corresponding command to turn in the correct direction. When it reaches the end of the line, it will send a message to reverse.

The beam sensors will be used to differentiate between power poles and nets based on height. There will be two beam sensors mounted on each side of the chassis, one at approximately 5” and the other pointing up aiming for approximately 18”. When both sensors on one side detect an object, it is identified as a power pole and the robot will continue moving. If only the lower sensor detects an object, the obstacle will be labeled as a net, and a message will be sent from the navigation system to the main microcontroller.

The main microcontroller will follow instructions directing the robot forward or backward until the shooting mechanism is properly lined up. When the mechanism is lined up, the navigation system will then send a ready to shoot command to the main microcontroller to allow the shooting mechanism to fire. After fired, the navigation system will proceed forward and repeat the process until all three nets are identified and beads are fired. At this point, the robot will return to the starting square and end the round.

The gap sensors will also be mounted on the side of the chassis and identify 4" gaps in the barricades. When a gap is identified, the navigation system will alert the main microcontroller which will then trigger the marshmallow subroutine to move the marshmallow off the roadway through the gap. While conceptually these sensors should work, issues may arise further down the line. Options for individual sensors and the overall system are continually being researched to find the best fit. After researching what sensors would be best, the team decided on the beam sensor for identifying nets, but are unsure of what complications may arise in testing relating to the netting material. Because the material is thin and not solid, most sensors will be unable to detect it, which is why discernment is taking place by pole height.

Due to the nature of the system, it will be primarily software, which is difficult to analytically test and measure. As long as used properly, predefined library functions will operate as expected, so the majority of testing will take place on communication between the microcontrollers. For testing purposes, signals sent from the navigation system will be measured and compared to signals received by the main microcontroller.

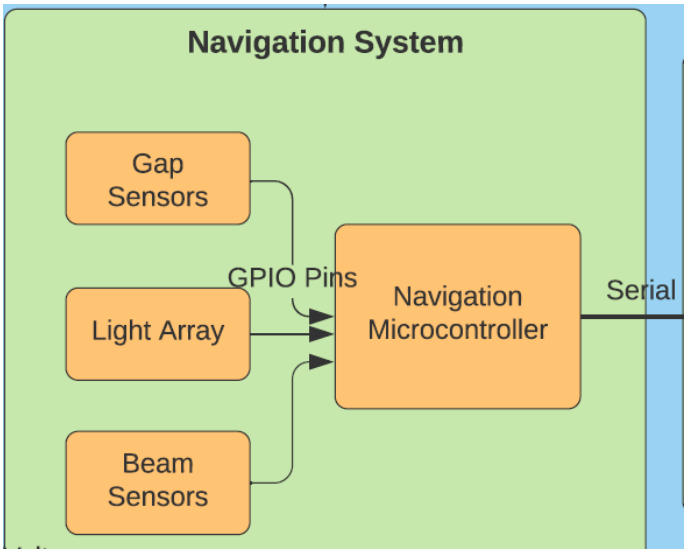

Fig. 6. Navigation block diagram

VI. Drive Train

This year, the environment that the robot will compete in is an L-shaped track. A large portion of the available points in the competition are rewarded solely based on the robot being able to navigate the track back and forth. As such, robot mobility is a critical component of the competition, and vital for the team's success.

The drivetrain subsystem of the robot (Figure 8) is responsible for moving the robot along the track. "Drivetrain" is an umbrella term that includes all parts that enable the robot to move through space, e.g., motors, wheels, gears, and so on. The drivetrain of this robot will be designed as a differential drive. A differential drive operates on the concept of moving the wheels at different speeds to enable steering. In total, the robot will have two DC motors and four wheels. The pair of wheels on either side of the robot will be connected to their respective motors via a system of gears, belts, or combination of both. One specification that the team will have to keep in mind is the size spec of the robot. The robot will need to be contained within 1 cubic foot which means that the drive train can be at most 1 square foot in area.

Selecting the right motor and understanding its performance characteristics is an important part of designing a drivetrain. A generic graph of DC motor performance is shown in Figure 7.

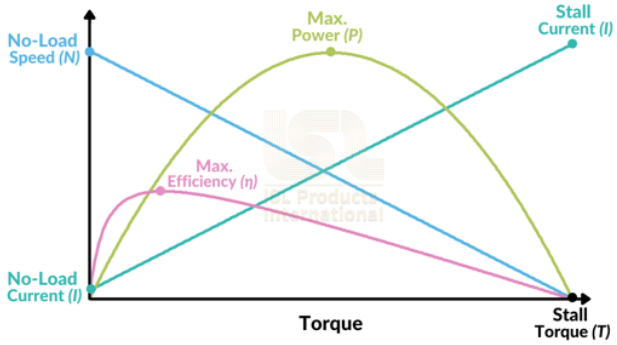

Fig. 7. Calculations of Arm

As can be seen in the figure, speed and torque of a motor are inversely proportional. As torque increases, speed gradually decreases. In addition, current and torque are directly proportional. At very fast speeds, the motor consumes little current and produces very little torque. This is an issue that can be alleviated with the use of gears. Attaching a gearset to the motor shaft will cause it to run at a lower speed, thus consuming more current and generating more torque. The amount of torque needed in this application is dependent on the weight of the robot and the desired acceleration. Given that the competition environment is a level track, it is not expected that a great amount of torque is needed. Only enough torque is needed such that the robot can accelerate at a reasonable rate and then maintain its velocity. For these reasons, it was decided that two DC motors would suffice. Using only two motors to drive the robot decreases the amount of complexity and needed parts when compared to driving four motors separately. In addition, the team will need to determine how fast the

maximum speed of the robot should be when navigating the track, and from there determine the gear ratio necessary to accomplish the task. Gear ratios work to linearly decrease the speed of the motor shaft. For example, a gear ratio of 2:1 means that the shaft will spin twice as slow and generate twice as much torque. Many DC motors come with predetermined gearboxes already attached to the shaft and are referred to as geared motors. Geared motors are a common solution in mobile robotics applications.

As the robot must be completely autonomous, this means that no member of the team will be allowed to do anything to affect the robot's performance after the run has started. Autonomous operation of the robot will require that the drivetrain system receives commands from the main microcontroller that detail speed and direction for the motors. The method of motor control will have to utilize feedback so that the commanded velocities are not random and reflect where the robot needs to go. One very common method of control is the Proportion, Integral, Derivative (PID) loop. This method of control is common in navigation applications. The proportion coefficient corresponds to how large the error is at any given time, the integral coefficient corresponds to the duration of the error, and the derivative coefficient corresponds to how quickly the error is changing. Using a combination of the three will allow the robot to navigate autonomously while maintaining the desired course.

PID controllers are only effective when they have been properly tuned and utilize the optimal coefficients. Methods of tuning a PID controller vary depending on the system in question. For an autonomous robotic application, a model-based approach may be appropriate. The electrical characteristics of DC motors are well understood, and they can essentially be modeled as a series RL load. This is advantageous for the engineers because a good starting point can be found even without hardware. One software that can be utilized to create a simulation is Simulink. In Simulink, the team would be able to create models of the system and test the responses. Graphs of the response can be created in Simulink that give a visual representation of the system response to error. One thing to consider when viewing these graphs is how much power is required to achieve the response. The amount of power will need to be feasible for our system. Any model-based tuning of the system would of course have to be manually tested by entering the determined PID coefficients into the motor controller and testing the navigation capabilities of the hardware itself. Once the coefficients have been entered into the hardware and the navigation performance has been tested, further tuning will likely be required in order to optimize the performance of the system. This will consist of changing each of the coefficients independently to see how it affects the system. The team will need to have a good understanding of how each coefficient affects the system so that this process relies less on trial and error. It is desirable that the system have a quick yet smooth response to error and that it is able to navigate the entire track while following the white line in the center.

The motor controller will generate Pulse-Width Modulation (PWM) signals to either a dual-motor H-bridge unit or two separate H-bridges that will provide power to the motors. PWM is a method of rapidly opening and closing a switch in a circuit that contains some main power source. The act of opening and closing the switch gives the engineer control over the output voltage to the load. A higher duty cycle (the percentage of the time the switch is closed) will result in a higher average voltage, and vice versa. PWM is advantageous to this project because it will allow the robot to spin wheels on either side of the unit at different speeds, giving the robot the ability to turn and navigate more easily. The team will need to determine a top speed for when the error is zero and the robot is driving directly along the line. All other commanded speeds will be calculated as a percentage of the top speed using the error and PID coefficients. The method of verification for the motor drivers will be LTSpice. An H-bridge is modelled as four transistors, two on each side of the motor. The direction of the current through the motor can be changed by biasing the different transistors, which in turn changes the direction of the motor. The PWM signals from the microcontroller will need to be simulated to ensure that they drive the H-bridges to give the correct average voltage to the motors. Once hardware is obtained, the team will use lab equipment such as the DMM to ensure that the voltage requirement is met.

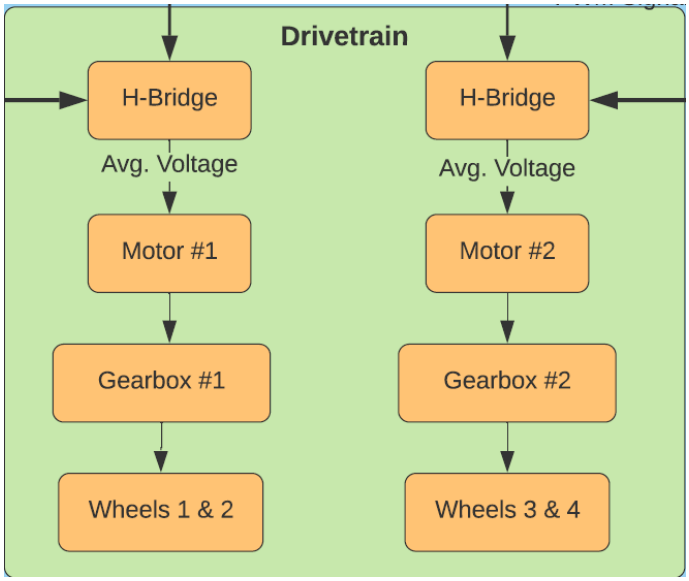

Fig. 8. Drivetrain block diagram

VII. Shooting Mechanism

The shooting mechanism (Figure 9) is a critical subsystem because of how the team has decided to maximize the number of points achieved. The shooting mechanism will allow for beads to be launched into vertical nets. This mechanism will be placed on the upper chassis, enabling it to shoot in either direction. The mechanism itself will consist of a belt and one motor, placed below the storage container on the chassis. When the main controller receives the shoot command from

the navigation system, it will enable the belt motor. At this point, the beads will be dropped onto the belt and accelerated to the speed required to reach the nets.

Prior to physical testing, modeling software will be used to test angle, velocity, and distance. Mathematical methods, such as projectile motion equations, will also be used for testing purposes. The team will need to ensure there is enough friction and speed on the belt for the beads to be appropriately launched. Belts with different materials and topologies will need to be tested.

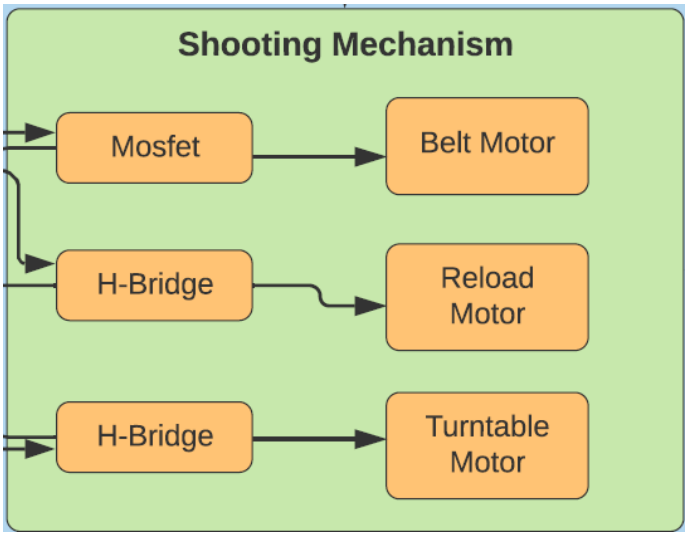

Fig. 9. Shooting mechanism block diagram

VIII. Arm

The robot arm (Figure 12) will receive power from the battery and inputs from the microcontroller. The microcontroller will send signals to servos 1-3 to position the robotic arm. Using the practice track, the arm will be preprogrammed to remove the beads from the trees. Once grabbed, the robotic arm will then receive a subroutine command to move the arm to the position to drop the bead/beads into the container to be shot. After this, the microcontroller will then command the servos to move to a home position. The robotic arm will be utilized again once the robot reaches the end of the track. The robotic arm design will need to be tested in many different aspects. For one, the feedback and functionality of the servos will need to be tested using Simulink. The team will also have to use analytical testing to find out the needed lengths of each section. The team will need to know the reach achieved by each section of the arm, which will then be added together to find out the end total reach achieved from the extended arms. The length of each section of the arm will be determined by the desired height and reach of the arm (Figure 10).

The team will also need to consider the grabbing mechanism length to add to the total distance achieved. The gameplay rules outline the specifications for the track [2]. The drawing can be seen in Figure 11. From the drawing, the team can see that there is a height window from the top of the branch

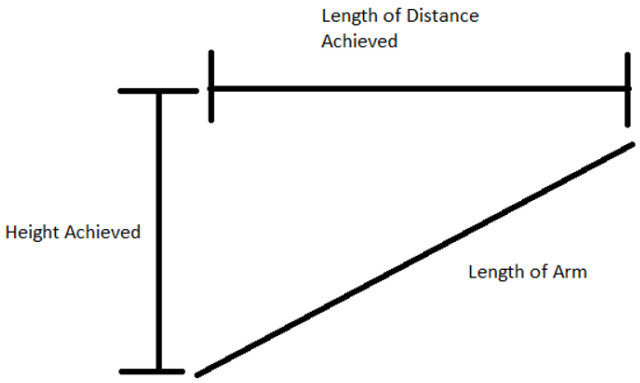

Fig. 10. Calculations of Arm

and to the bottom of the hanging beads of 6”- 9” from the top of the robot. The team can also see that there is a length window from the front of the branch to the back of the branch ranging 6” - 12” from the side of the robot. The combination of the analysis plus the gameplay rules outline will determine the lengths of the arms.

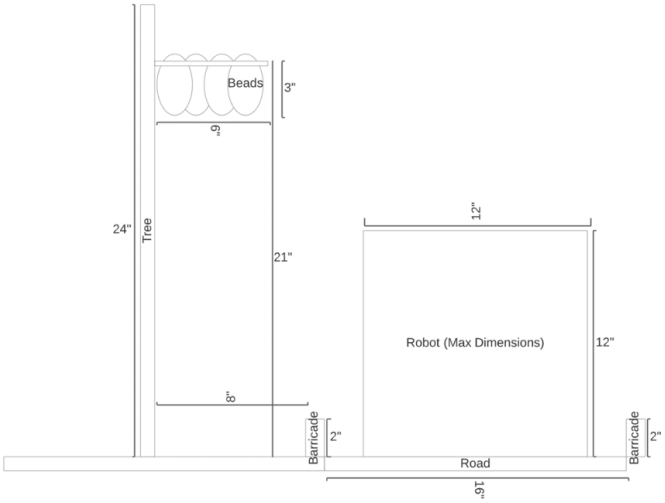

Fig. 11. Gameboard specifications

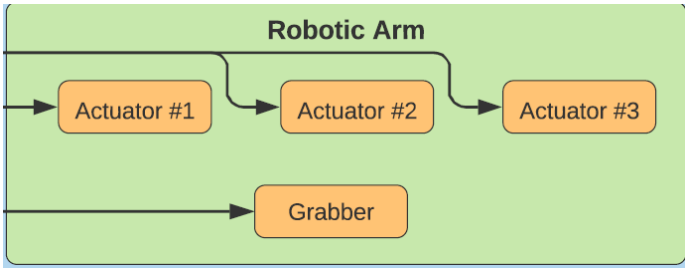

Fig. 12. Arm block diagram

IX. Peripherals

The peripheral system (Figure 13) is responsible for controlling the visual and aesthetic elements of the robot in order to receive the maximum number of competition points. Because of the non-critical nature of such a system, the peripherals will not communicate or receive any inputs from the other subsystems except for a constant voltage from the power supply system. Furthermore, this system will be broken into two distinct elements. The first element will be an analog circuit responsible for lighting the various LEDs that will decorate the robot. This will be a simple circuit that connects the LEDs to the power supply and pull-down resistors. The second element of the peripheral system will be a small microcontroller responsible for powering and controlling both the LCD display and a speaker via separate functions. For the LCD display, the function will be responsible for displaying a hard-coded image/set of images and text, and for the speaker, a preselected song will be played using either the microcontroller's GPIO or DAC pins.

In order to verify the operation of the peripheral system, an LTSpice simulation of the analog LED circuitry will be required. This analysis will be used to determine the ideal number and configuration of the LEDs, as well as the ideal component values of the elements of the circuit that will make the LEDs flash or blink. Additionally, if the DAC of the microcontroller is to be used for the speaker, further LTSpice testing must be conducted to test whatever amplification circuitry, if any, that will exist between the DAC and the speaker itself. Lastly, the output signals of the microcontroller must be experimentally verified to ensure that the correct waveforms are being generated before they are sent to the external hardware.

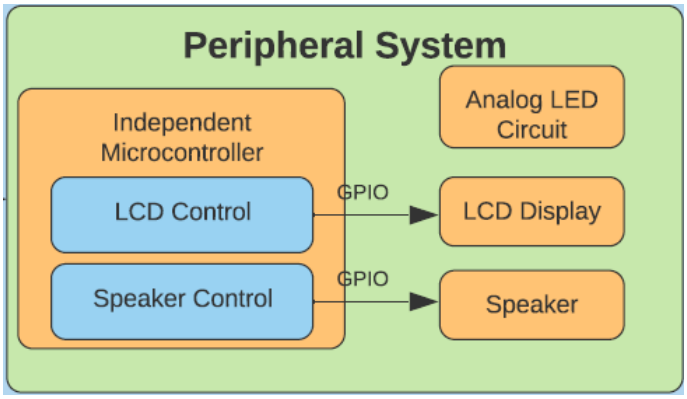

Fig. 13. Peripherals block diagram

X. Architectural Risk Analysis

It is also vital to consider the inherent cybersecurity risks associated with any proposed design. This Architectural Risk Analysis (ARA) is critical in order to develop a solution that considers and mitigates the possible threats and vulnerabilities of the system. The nature of this design's purpose presents a unique angle from which risks must be assessed and addressed. Both the nature and rules of the competition apply a heavy influence on the overall ARA of the design.

Because of the nature of the competition, it is expected that other contestants will make attempts to attack and impede the design. The most likely attack scenario will involve the variety of sensors on the exterior of the robot. The communication between the sensors and the microcontrollers will be contained within the confines of the robot itself and will not utilize any wireless communication. Therefore, the most likely method of attack will be directed at the measurements taken by the sensors themselves. For example, a bright light could be used to disrupt the infrared sensors and cameras. In order to mitigate such risks, extensive experimental testing in a variety of environments and conditions must be conducted prior to the competition to determine the risk of such attacks and to properly adjust the sensors to be more resilient to nonideal conditions.

While the competition presents a higher chance of attack, one of its most critical aspects also minimizes or eliminates the majority of the design's attack surface. The competition rules state that the robot must be completely autonomous and that team members must verify this by suspending any outbound communication from any personal devices (laptops, cell phones, etc.). Because of this stipulation, the overall attack surface is minimized by design. Since no inbound communication is expected by the robot, there are also no vectors for attackers to utilize to penetrate the system. The robot will function as a fully isolated system, so no opportunities for attack will present themselves by design. Also, it is safe to assume that outbound communication from competitors' devices will also be restricted, as they are also actively competing by the same rules. Because of this, the ability to attack the system will be minimized because competitors will not have the means to attack the system, similar to how the team will not have the means to assist the system. These design choices and stipulations lead the team to conclude that the attack surface of the design is effectively zero, and that only environmental effects on the sensors could jeopardize the design. These will be mitigated via extensive testing beforehand.

XI. Gantt Chart

The Gantt chart was laid out in a way to ensure that each team member would not be overloaded with multiple tasks within the same week. In saying this, each member has been distributed evenly so that every task is completed by the deadline. The tasks are also laid out in such a way that the basic task would get completed and then built on with more advanced tasks. For example, the shooting mechanism bead trajectory calculations are shown to be completed before the preliminary testing or model are completed. In this instance, the team needs to know the desired speed and distance the bead needs before knowing if the launching mechanism can achieve the distance. Similarly, the microcontroller preliminary testing needs to be completed before the team can model the system. Outlining the tasks in this way will help ensure the

team is able to complete all tasks before the required deadline. The chart is attached in Figure 15.

XII. REFERENCES

[1] “Arduino Mega 2560 Rev 3,” arduino.cc, [Online]. Available: https://store.arduino.cc/products/arduino-mega-2560-rev3. [Accessed: Oct. 17, 2021].

[2] IEEE SoutheastCon, “2022 SoutheastCon Hardware Competition Rules,” ieeesoutheastcon.org, Sep. 3, 2021. [Online]. Available: https://ieeesoutheastcon.org/wp-content/uploads/sites/309/2022_SoutheastCon_HardwareRules-Final.pdf [Accessed: Oct. 17, 2021].

[3]J. Gallant, “How to read DC motor & gear motor performance curves,” ISL Products International. [Online]. Available: https://islproducts.com/design-note/how-to-read-dc-motor-gear-motor-performance-curves/. [Accessed: Oct. 31, 2021].

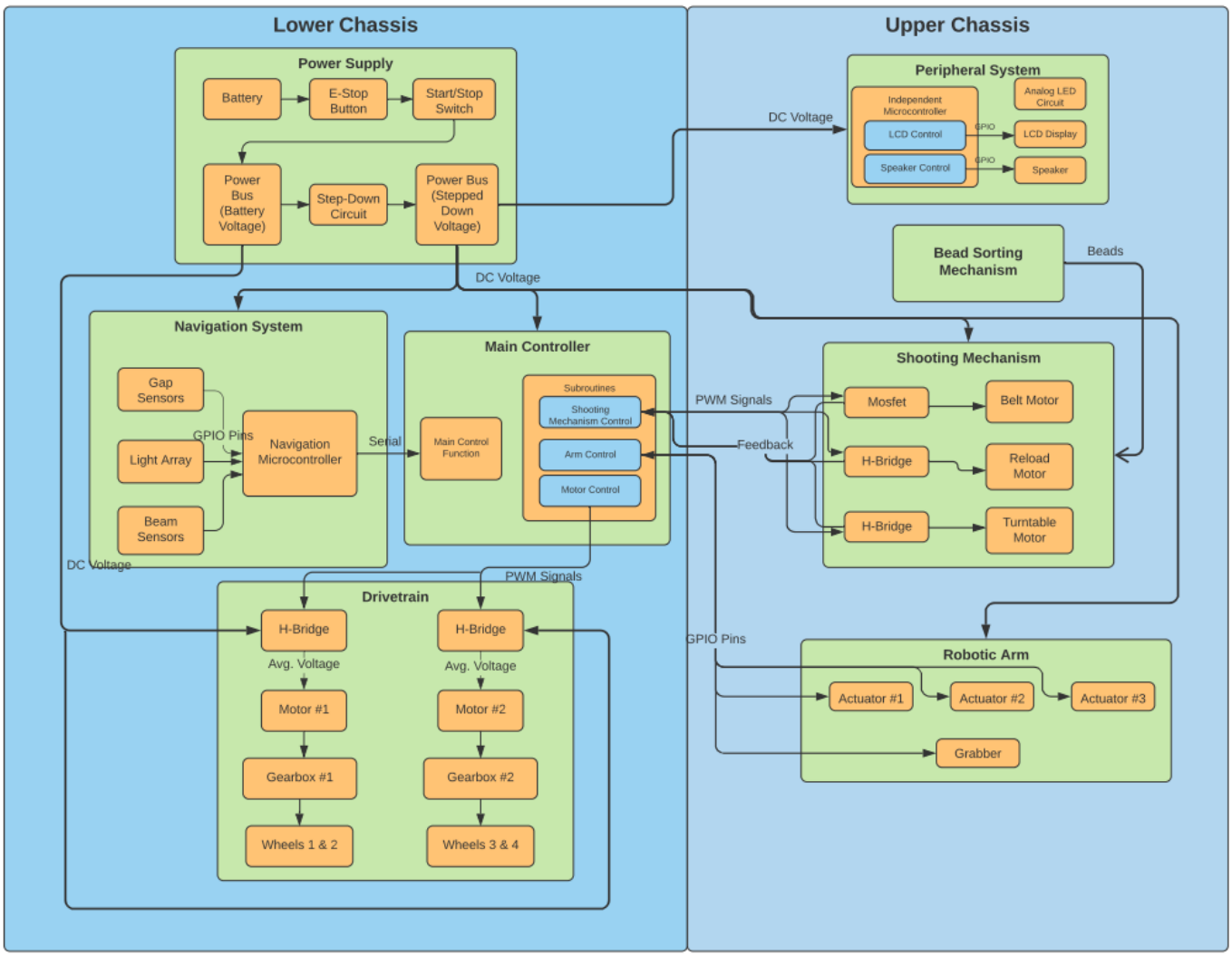

Fig. 14. Fulll block diagram

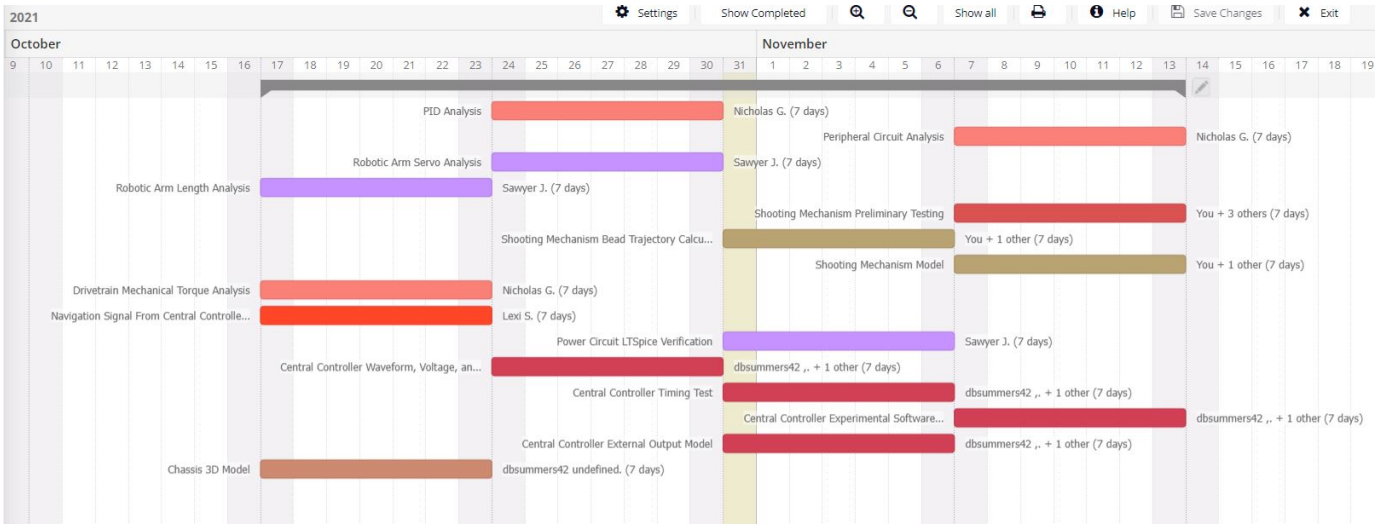

Fig. 15. Gantt chart