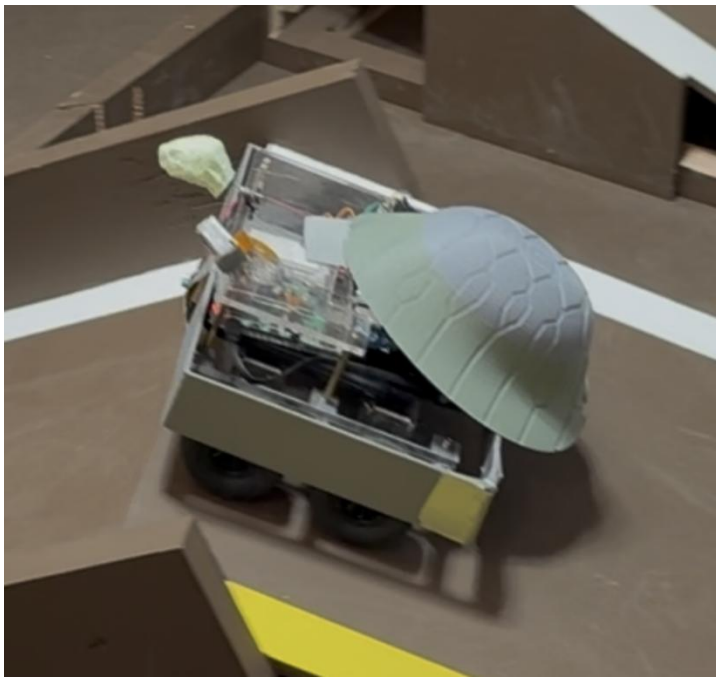
Turbo Turtles Autonomous Robot

MCEN 4115-5115: Mechatronics



December 9, 2025

Instructor:
Derek Reamon

Team Members:
Dominic Snyder
Sean Campbell
Bill Hsu
Jack McGuffee
Luke Jensen
Simon Holstein
Jay Warren

Executive Summary

The goal of this project was to design and build a robot capable of autonomously navigating a course, with optional functionality for collecting and sorting colored dice. The primary objectives were to achieve reliable autonomous navigation, withstand stresses during use, consistently traverse a gravel pit, and remain within the dimensional requirements of the competition. A secondary stretch goal was to develop a dice pickup and sorting system.

To meet these goals, the robot was divided into four subsystems: the drive system, dice sorting system, navigation system, and the electrical system. The team followed an iterative development approach, implementing improvements across several prototypes to converge on a final design.

The drive system comprised an acrylic chassis, four independently driven DC motors, and custom wheel shafts supported by bearings to reduce loading of the motor shaft. High-traction wheels and increased motor speed enabled the robot to traverse the gravel pit reliably.

The navigation system utilized a Raspberry Pi, Arduino, a wide-angle camera, and a custom Python line-following algorithm. The system converted raw camera images to HSV, detected the course boundary, calculated the line centroid, and used proportional control to steer the robot. Additional maneuvers allowed the robot to recover when off-track or stuck. The navigation approach proved effective and robust after tuning.

In parallel, a dice pickup and sorting mechanism was developed using 3D-printed rollers, rubber bands, custom gearing, and a hinged mounting system. Although functional, this subsystem added significant weight and shifted the robot's center of mass, preventing successful traversal of the gravel pit. As a result, it was removed for the final competition.

Overall, the final integrated robot successfully navigated the course and demonstrated consistent autonomous behavior. Key lessons learned included the importance of early subsystem integration, designing mechanical components with potential failure modes in mind, building adjustability into prototypes, and favoring simple solutions over unnecessarily complex ones. These insights informed both the final design and our approach to future engineering projects.

Table of Contents:

Executive Summary 1
1 – System Overview and Requirements 3
2 – Design Process 4
3 – Drive System 4
3.1 – Chassis 4
3.2 – Motors 5
3.3 – Wheels and Shafts 6
4 – Dice Sorting System 6
5 – Navigation System 10
5.1 – Sensing 10
5.2 – Control Algorithm 11
5.3 – Notes 12
5.4 – Code 12
6 – Electrical System 12
6.1 – Notes 13
7 – Conclusion 14
8 – Appendix 15
8.1 – Electronic Diagrams 15
8.2 – Racecourse 17

Table of Figures:

Figure 1: Turbo Turtles Robot 3
Figure 2: Initial Chassis Prototype. 5
Figure 3: Wheel Shaft 6
Figure 4: Roller Mechanism. 6
Figure 5: Three Tier Roller Mechanism. 7
Figure 6: 60-RPM Motor & Geartrain. A) Side View. B) Front View. C) Pin Glued to Roller. 7
Figure 7: Fully Integrated Robot (Minus a Wheel) 8
Figure 8: Dice Sorter Mechanism. A) Bottom View. B) Top View. 8
Figure 9: Dice Collection Funnel. 9
Figure 10: Dice Collection Hinge Mount. 9
Figure 11: Early CAD Prototype Highlighting Initial Dice Collection Mechanism. 10
Figure 12: A) First Protoboard. B) Second Protoboard. 13
Figure 13: Final Turbo Turtle used in Competition. 14

1 – System Overview and Requirements

The project had several goals that dictated our system requirements. The main goal was to design a robot that could autonomously navigate a course (course diagram in Appendix), with bonuses awarded for collecting and sorting colored dice. Given the provided specifications of the course and challenge, we defined the following requirements for the robot:

1. The robot shall autonomously navigate the defined course.
2. The robot shall withstand any forces and impacts experience during the race without failure.
3. The robot shall be capable of traversing a gravel pit.
4. The robot shall include a rigid barrier as the outermost structure, extending 2” to 4” from the ground.

While not required to be attempted during the competition, we also set the goal of developing an effective dice collection and sorting mechanism.

The robot was divided into four sub-systems to structure the design and effectively delegate tasks. The sub-systems are as follows:

- Drive System
- Dice Sorting System
- Navigation System
- Electrical System

Technical details on each sub-system are described in the following sections.

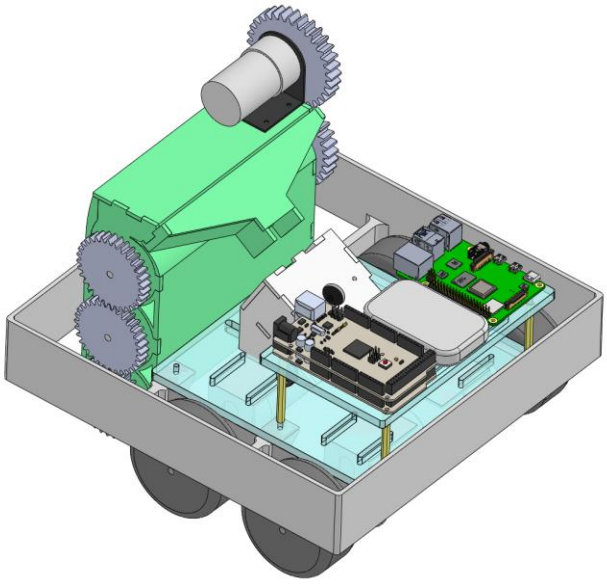


Figure 1: Turbo Turtles Robot

2 – Design Process

The team used an iterative approach to the design process of the Turbo Turtle Robot, allowing rapid prototyping and continuous refinement. The project timeline followed the course labs, which outlined when each subsystem needed to be implemented.

During initial concept development, the team collaboratively brainstormed and decided on the general specifications for each subsystem. This included decisions regarding what type of drive system was to be used, sensor selection, navigation method, and an overall approach of collecting and sorting dice. Responsibilities for each subsystem were then divided based on these definitions and the members' interests and skills.

Initial prototypes aided the design and allowed refinement as the development continued. The first subsystem with a working prototype was the chassis and drive system to allow the navigation team to start writing and testing sensors and code. In parallel, the dice sorting team iterated through designs for both the sorting system and collection system.

Final integration of all subsystems happened once all the systems were ready for installation. During the final week of testing and final refinements, the dice collecting system presented challenges. As a result, the final iteration of the robot focused on autonomous navigation aiming to provide good track time during the competition.

The following sections outline the specific design choices and details for each subsystem.

3 – Drive System

3.1 – Chassis

The chassis was designed to provide adequate support for all the components and withstand the expected forces during use. At the same time, minimizing its weight was a primary consideration to minimize stresses and enable traversing the gravel pit.

For the main body, we chose to design custom acrylic plates as they provided adequate strength, low weight, and were easy to manufacture. The manufacturing method we chose was laser cutting because it was efficient and offered tolerances within our specifications.

Our initial design used 1/8" thick acrylic, however, testing revealed that the plate bent when fully loaded, causing the wheels to move out of alignment. This led us to switch to 1/4" thickness. Mounting holes were added to the secure all the necessary components and slots were used for the motor mounts to allow for adjustability.

A second plate was included and separated from the main plate by spacers to provide a mounting point for the camera and additional space for electronics.

Side supports were mounted to either side of the main plate along its entire length. The side supports both added rigidity and housed bearings for each wheel shaft. The bearings supported the wheel shafts, and in turn the wheel, to reduce the bending force on the motor shaft.

M3 fasteners were used throughout the robot to fasten components. Where necessary, flat washers and lock washers were used to prevent loosening from vibration. This approach provided a reliable method of fastening that was removable, which was ideal for iterating and troubleshooting.

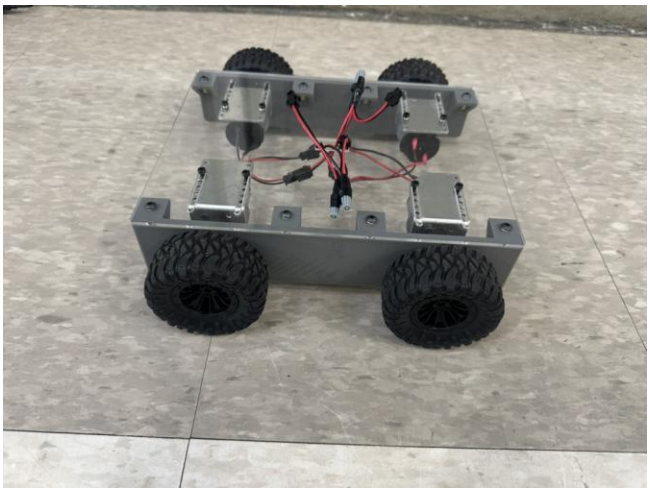


Figure 2: Initial Chassis Prototype.

3.2 – Motors

Our approach for power and steering was to use 12V DC motors and utilize “tank” steering by providing power to each wheel in varying ratios to turn the robot. After several iterations, we decided to mount a motor at each wheel to improve efficiency and minimize the number of mechanical components required.

Our initial prototype included 100 RPM DC motors, but we switched to 200 RPM motors for the final version to provide higher speed. We calculated the required torque for the motors using the diameter of the wheels, the weight of the bot, the expected coefficient of friction, and efficiency.

A hole was machined through each motor shaft so that it could be coupled to the wheel shaft with a fastener. This ensured that the wheel would not separate from the motor during use.

The motors were attached to the main plate via motor mounts.

3.3 – Wheels and Shafts

We selected 3” diameter wheels with significant tread to provide the clearance necessary to navigate the gravel pit. They were connected to the motor via a custom wheel shaft, shown in Figure 3.

One end of the wheel shaft coupled to the D-shaft of the motor, with a hole where a fastener was secured. It then passed through the bearing in the side support, where the other end was a hex that inserted into the wheel. This side was secured to the wheel with an M1 fastener to ensure it didn’t come loose.

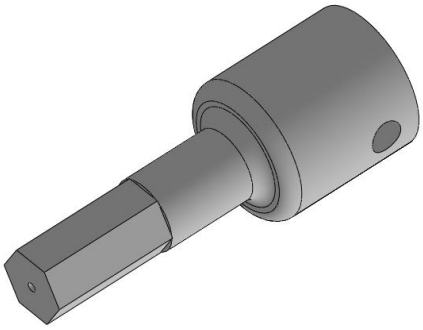


Figure 3: Wheel Shaft

4 – Dice Sorting System

The dice sorting system was split into two separate subsystems: pickup and sorting. For the pickup mechanism, we spent a significant amount of time prototyping different ideas to obtain the dice from the ground. Initially, we considered a brush design, like a vacuum carpet brush roller, that would roll the dice up into our sorting mechanism. However, after some three-dimensional design iterations on CAD, we landed on a roller that used rubber bands instead of brushes. The rubber bands were meant to flex around the dice and utilize the friction between the dice and the rubber to roll the dice up a ramp.

Figure 4: Roller Mechanism.

Continuing with this idea, we needed to maneuver the dice from the ground onto our robot for sorting, so the issue of height came into play. With only one roller and a ramp, we could only lift the dice up a few inches. Furthermore, we could not use a larger diameter roller than roughly 2.5 inches due to the size constraints of the robot. Thus, we designed a mechanism involving three tiers of these rollers (seen in Figure 5), with walls on either side of the roller that matched the round contour and allowed full contact between the dice, the roller, and the wall. This design collected the dice from the ground and transported them up to around 8 inches, moving from roller to roller in a snake-like path until it reached the top of this mechanism for sorting.

Figure 5: Three Tier Roller Mechanism.

To rotate each of these rollers, a 60-rpm electric motor was used in conjunction with a geartrain system (seen in Figure 6). In this system, a gear was attached to the motor and each roller via a 5-millimeter pin. These gears then meshed to allow the use of only one motor. Similarly, the way these gears meshed allowed for the top two rollers to rotate in the same direction while the middle roller rotated the opposite direction, which was necessary to move the dice around the three rollers in a snake-like path to the top. Below are pictures of the gearing mechanisms and the final assembled bot (with one wheel missing due to a collision) and sorting mechanism.

Figure 6: 60-RPM Motor & Geartrain. A) Side View. B) Front View. C) Pin Glued to Roller.

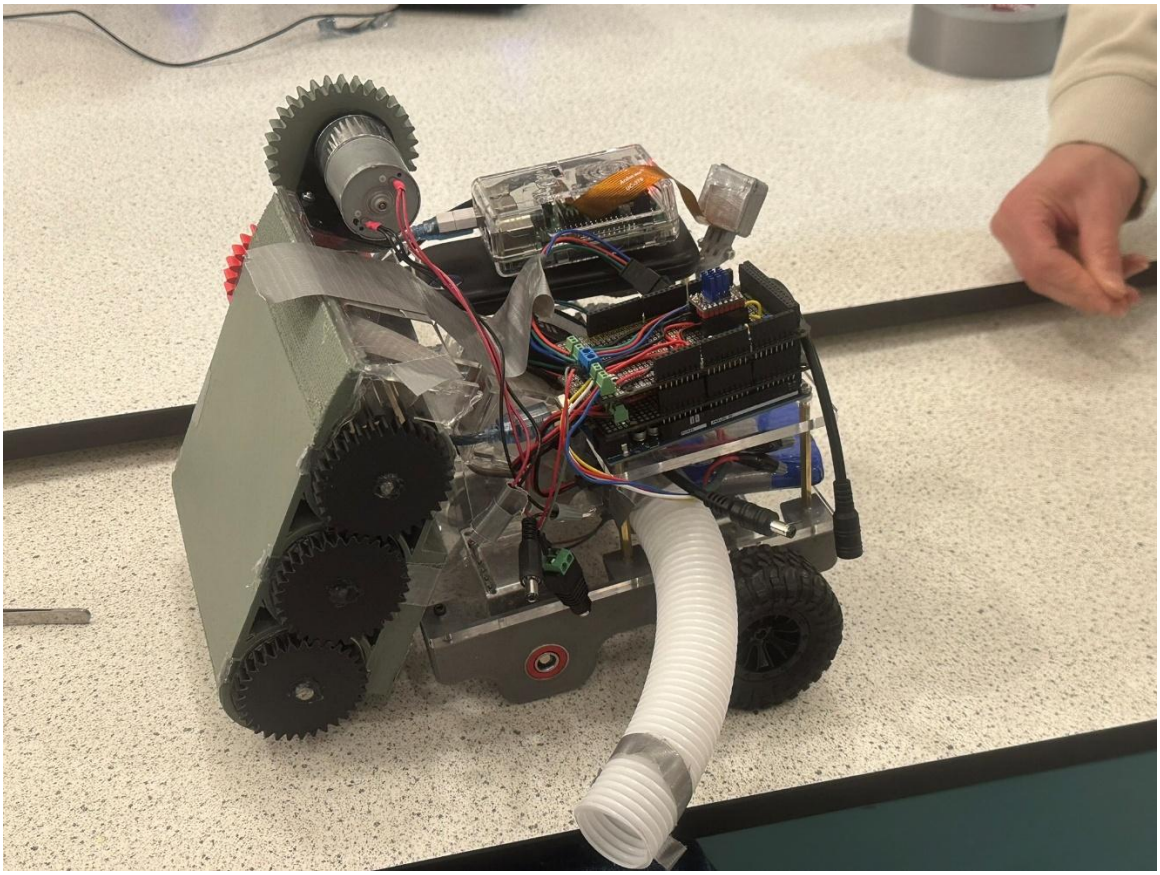


Figure 7: Fully Integrated Robot (Minus a Wheel).

To sort the dice based on color, a funnel system was used to transport the dice from the pickup mechanism into our sorter. A color sensor connected to the Arduino board detected green or red dice and either turned the sorter left or right using a stepper motor to dump the dice either out of the robot (if the wrong color) or into a holding area. Below are images of the sorter.

Figure 8: Dice Sorter Mechanism. A) Bottom View. B) Top View.

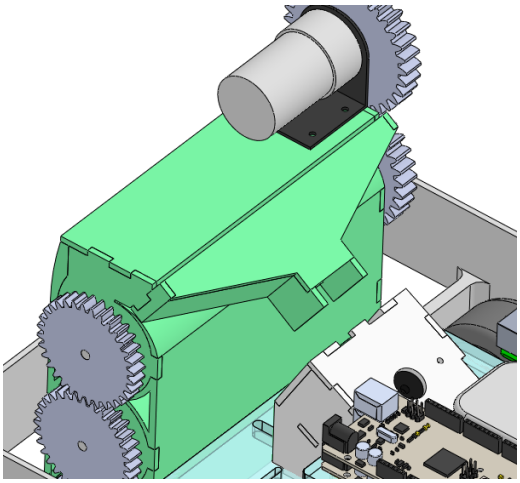


Figure 9: Dice Collection Funnel.

Lastly, to implement this entire dice sorting system onto the robot, we utilized a standard door hinge with a return spring. This component was meant to achieve two goals:

- Attach the pickup mechanism to the robot securely
- Allow for some rotational movement of the pickup mechanism. This was needed for two reasons:
    - Since the pickup mechanism only hovered about a millimeter above the ground during operation of the robot, it would drag on the gravel pit. This hinge allowed the pickup mechanism to rotate out of the way if it contacted any rocks.
    - When the robot drove up or down a ramp, the pickup mechanism would drag against the ground and stall the robot. This hinge would allow the mechanism to rotate slightly so all four wheels of the robot could contact the ground and provide torque to overcome the ramp transitions.

Slots were incorporated on the back of the chassis to mount the pickup mechanism, and the hinge served as the connection between the two components (chassis and pickup mechanism).

Figure 10: Dice Collection Hinge Mount.

The dice sorting system proved to be a very challenging engineering problem, requiring several iterations and prototypes before we landed on a functional design. We utilized 3D printing and laser cutting to create most of the parts, which allowed for easy but time-consuming prototyping. Our final design was constructed of mostly PLA and acrylic, except for parts like the motor, pins, hinge, and rubber bands.

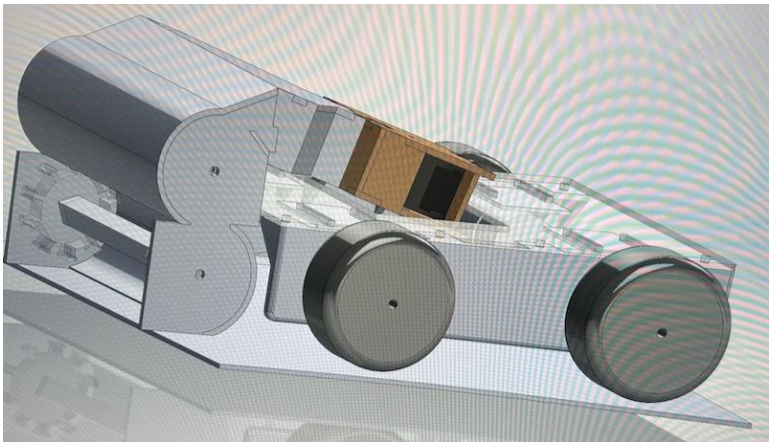


Figure 11: Early CAD Prototype Highlighting Initial Dice Collection Mechanism.

An earlier design of our sorting system is shown above. We found that the height of this system did not provide enough vertical height to move dice downward into the sorter.

Ultimately this sorting system was omitted from the robot for the competition due to its heavy weight and bulkiness. Since the robot's wheelbase was relatively short, the heavy pickup mechanism on the back of it prohibited the robot from overcoming the gravel pit: a key obstacle to overcome if we wanted to be successful in the competition.

5 – Navigation System

5.1 – Sensing

We relied entirely on a Arducam for Raspberry Pi Camera Module 3 Wide, 120°(D) IMX708 Autofocus Pi for line following. We also had a single front button mounted on the front bumper to detect collisions. Our control logic was implemented in Python on the Raspberry Pi which sent motor commands to our Arduino (via a serial cable) which controlled all motors and relayed button pushes back to the Pi. While we did not use a sorter on our final robot, our initial dice sorting system was entirely implemented on the Arduino which received input from a color sensor.

We experimented with putting buttons on the bottom of the bot to detect the gravel pit and engage motors at full speed; however, we found that they caught on the rocks and were ultimately not necessary after reducing the weight of the bot.

5.2 – Control Algorithm

We implemented a simple but effective line following algorithm using raw camera images as follows (the code linked below may be shorter and easier to follow than this description):

- **Color Space Conversion:** The raw image color channels are converted from BGR to HSV to improve color segmentation robustness against lighting changes.
- **Thresholding & Cleaning:** A binary mask is generated by thresholding for a specific yellow color range. OpenCV morphological operations (opening and closing) are applied to remove noise and fill gaps in the detected line. This morphological step is likely not necessary as our mask was pretty robust during testing.
- **Scanline Analysis:** Rather than analyzing the entire frame, the algorithm extracts the horizontal row at the midpoint of the image.
- **Centroid Calculation:** The algorithm calculates the mean x-coordinate of all non-zero pixels within that scanline. This value represents the detected line position.
- **Offset calculation:** We define a fixed offset that we want the centroid value to equal. This had to be determined manually by placing the robot on the track and seeing what the centroid values were when the robot was approximately 2 inches from the yellow wall. This number was used as our offset. It's worth noting that this number would change if the camera position was disturbed, which caused us several misadventures during the competition.
- **Proportional control on the offset:** We implemented a simple proportional control algorithm which increased the robot's drift to the left when the difference between the centroid and offset was negative and increased the robot's drift to the right when the difference was positive
    - **Hard right turn when offset is very large:** If the offset is very large, the robot is likely approaching a horizontal yellow line. We continually execute a right turn while the offset is above some "large" threshold.
    - **Slight turn to left if offset is missing for a short period:** If we cannot see the line, we drift left.
        - **If the left line has not been seen for at least 3 seconds**, we spin in place while looking for a yellow line in the top 10% of the image (rather than the midpoint). Once found, we move slightly forward.
        - **If the left line has not been seen for at least 12 seconds**, we assume that we are stuck and execute a **"go insane"** maneuver which has the robot move forward at full speed then backward and then left.
- **Front bumper hit**: If the button on the front bumper of the robot is pushed, we assume we are stuck on a wall and execute a preprogrammed move backward and turn left maneuver.

5.3 – Notes

We ran our bot at about 60% of max speed because it made it easier for the control algorithm to recover from mistakes and to keep the robot from flipping over when going up/down the bridge due to weight distribution issues.

While our system was robust and simple, it suffered from 2 main problems:

1. It would frequently lose and regain sight of the line while going up the bridge due to our bot losing traction on its front wheels. This often caused it to wobble left and right until it fell off the bridge.
2. The algorithm was tuned to a specific camera position. If the camera position was sufficiently altered, the bot struggled to navigate.

5.4 – Code

Our python Raspberry Pi controller code and Arduino motor driver code can be found on our project GitHub.

6 – Electrical System

The diagrams in the Appendix depict the overall electrical system of our self-driving robot (not pictured are the Raspberry Pi and camera). The Arduino mega is used to control the sorting stepper motor and drive motors, read color sensor readings, and read commands from the Raspberry Pi. Specifically, the TMC2209 stepper controller and CS3200 color detector are used to sort dice. The TB6612 duo channel motor driver is used to drive four DC motors. Two motors are connected to one side of the bot in parallel as a 4-wheel drive system. One additional motor is powered separately to rotate the pick-up-dice mechanism. Two spare buttons are placed on the front of the robot to detect collisions. They are triggered if the robot makes contact with something on its front bumper. Ultimately, we only ended up using one button, installed in middle front of the bot for collision detection.

Two Arduino Mega protoboards are used as the main control board for the system. The first protoboard (Figure 12.A) is used to connect the TB6612 motor driver to the Arduino Mega. The second protoboard (Figure 12.B) is used to connect the TMC2209 stepper controller and seed studio CS3200 color sensor. They are stacked to be compact.

22awg UL1007 solid core wires and several screw terminals are used for the wiring of the protoboards. Blue large terminals are used for powering the bot and green small terminals are used for signals. A 2.1 DC Jack is extended from the power screw terminal to connect the 12V battery, which powers the whole system. 5V is generated by the Arduino Mega LDO itself and used to support all systems. 4-pin and 2-pin JST SM connectors are applied for motors and sensors connection. A USB cable is used to connect the Arduino and Raspberry Pi for self-

control and navigation, which is not demonstrated in the schematics. The soldered protoboards are demonstrated in Figure 12 below.

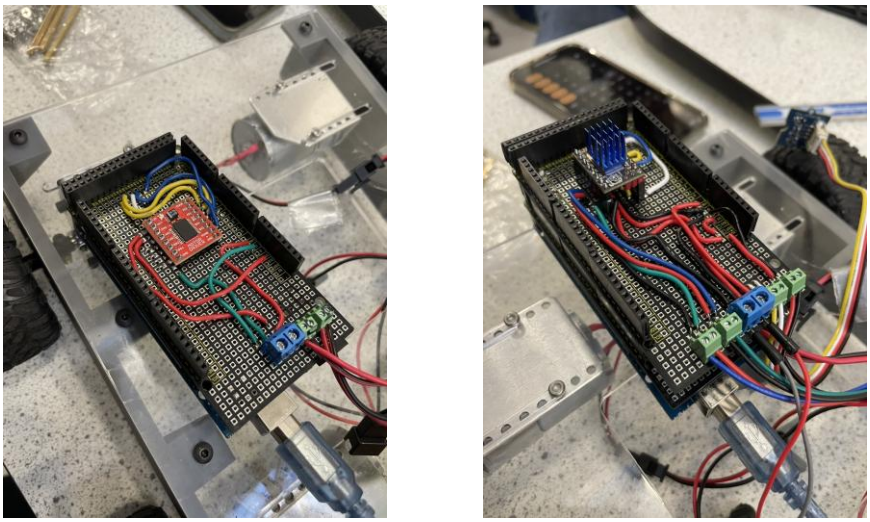


Figure 12: A) First Protoboard. B) Second Protoboard.

6.1 – Notes

For the final contest, we decided to not use the sorting system and so we only used the motor driver protoboard. **Unfortunately, we fried our original motor driver protoboard by switching power and ground while trying to find a loose power connection and then fried our spare motor driver protoboard (in an unknown way) the morning of the contest day**. Thus, we had to frantically rewire the motors using a breadboard right before the competition began.

We dug deeper after the final contest, trying to find out the reason why the second motor driver on the protoboard was fried. We suspect one motor connector's housing was loose, causing power wires to short circuit. Most of the housing was solid; however, we found a loose connector that we had removed during earlier development to fix a wiring mistake. We believe this likely caused a catastrophic short circuit.

7 – Conclusion

This project was a useful learning experience for the team, and we developed many new skills along the way. We came away with a few key lessons:

1. It is important to get to independently functional sub-systems as quickly as possible. Although the dice pick up mechanism was functional, we ran into trouble balancing the weight and clearing the gravel pit once it was added to the chassis. We didn't have enough time to troubleshoot these issues, so we couldn't use it for the challenge. If we had integrated sooner, we would have had more time to fine tune the system.
2. Consider potential failure modes of mechanical components throughout the design. The initial design of the wheel shaft had a stress concentration, so it broke during testing and required a last-minute redesign. Thinking more critically about the component's requirements and conducting more thorough testing could have prevented this.
3. Allow room for iteration and adjustment in early prototypes. Some examples we encountered included using slots instead of holes, fasteners instead of glue, and soldering header pins to connect motor drivers instead of soldering them directly. Approaches such as these account for the inevitable adjustments that will need to be made during testing.
4. Start simple and only add complexity if truly necessary. Many of our best design decisions, such as using a single camera to navigate, came down to using the simplest possible solution that worked. This minimized the number of factors to troubleshoot, which was key given our timeline. Adding features unnecessarily increases the potential failure modes without providing additional benefits. We found that starting with a minimum viable product and adding complexity only when required led to a more streamlined design.
5. Below is an image of our final robot used in the competition. We had to dramatically modify it from the original design.

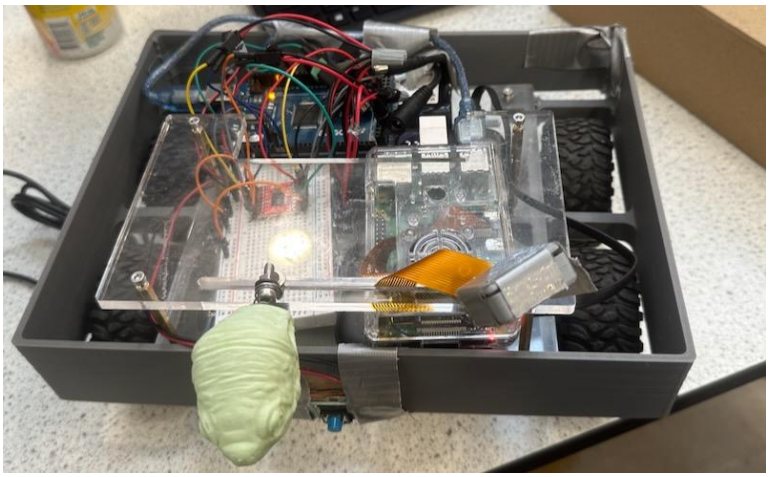
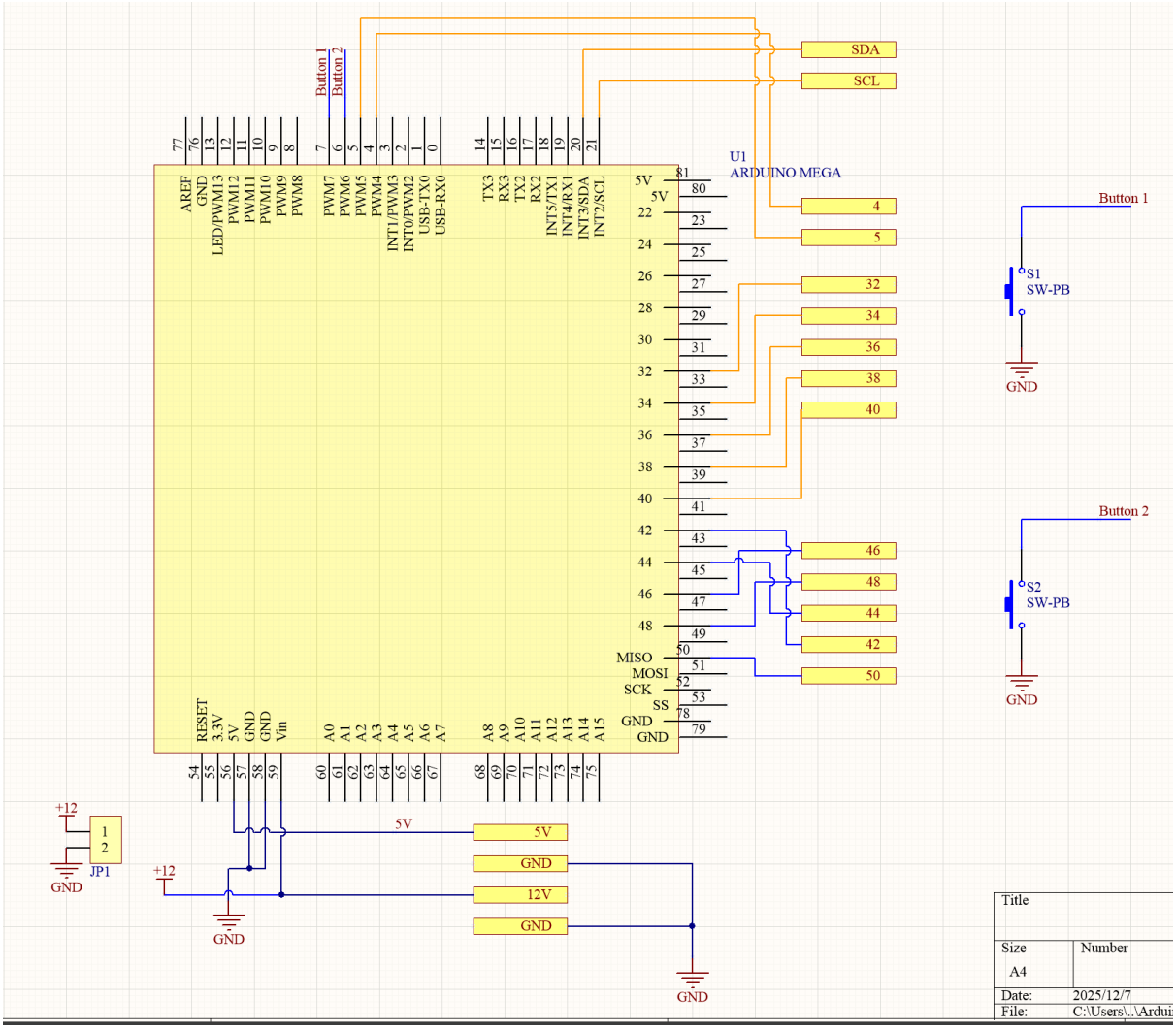
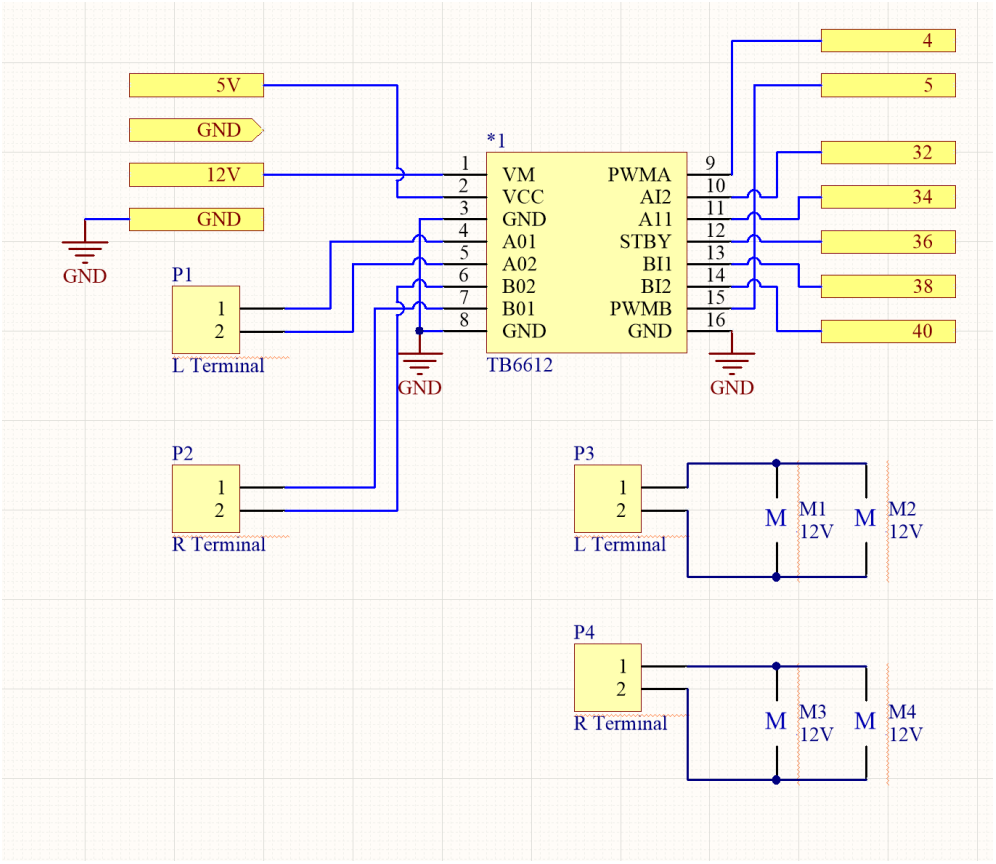


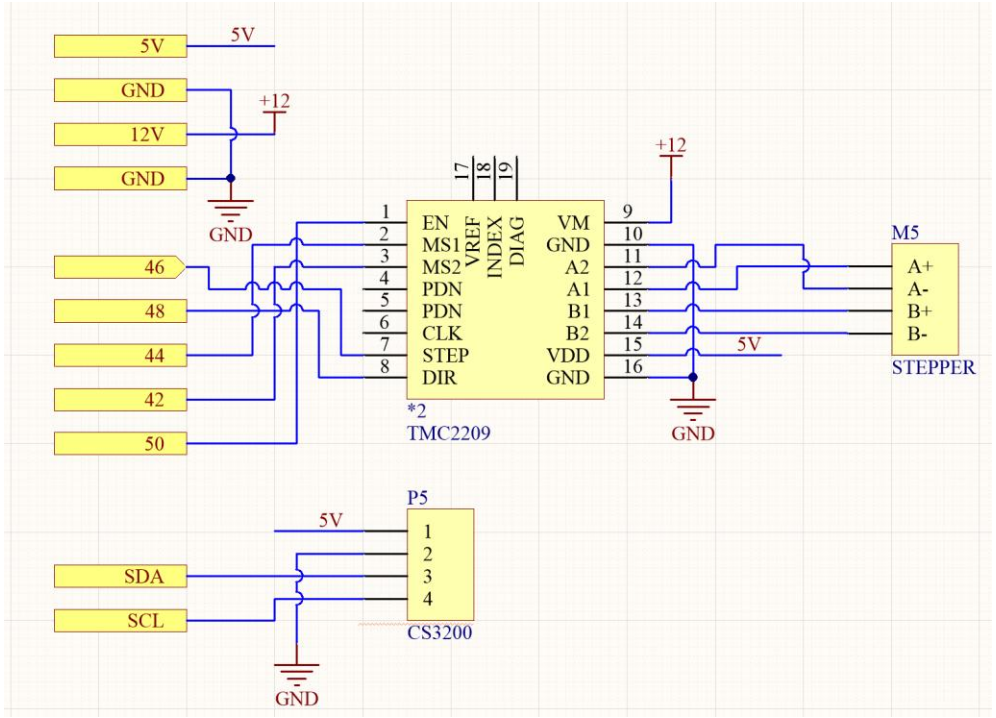
Figure 13: Final Turbo Turtle used in Competition.

8 – Appendix

8.1 – Electronic Diagrams




5V
GND
12V
GND
GND
*1
VM
VCC
GND
A01
A02
B02
B01
GND
PWMA
AI2
A11
STBY
BI1
BI2
PWMB
GND
TB6612
4
5
32
34
36
38
40
P1
L Terminal
P2
R Terminal
P3
L Terminal
M1
12V
M2
12V
P4
R Terminal
M3
12V
M4
12V


5V
GND
12V
GND
+12
46
48
44
42
50
EN
MS1
MS2
PDN
PDN
CLK
STEP
DIR
VREF
INDEX
DIAG
VM
GND
A2
A1
B1
B2
VDD
GND
*2
TMC2209
M5
A+
A-
B+
B-
STEPPER
P5
SDA
SCL
CS3200

8.2 – Racecourse