ENGR2781 – Mechanical Design Project

Design Report – R2-SPIN2

1. Executive Summary

The 2025 Warman design and build competition involves solving a mock issue where three meteorites must be moved across an unstable bridge and to a designated fuel bunker deposit zone. The people of Gondwana have expressed concern to Earth's engineers as their resources for energy are depleted, however, due to recent events three meteorites have landed on the planet of Gondwana which serve as a viable replacement for its people's energy.

To aid this community, Earths engineers were tasked to create an autonomous robot capable of collecting, storing, transporting, and depositing these meteorites to a fuel bunker where their energy can be extracted. To simulate this, a scale model was constructed and programmed to run through a course. To simulate the meteorite collection a 3x3 grid contained three varying sized balls that were randomly placed, further, to simulate the unstable bridge a large seesaw was constructed, and finally to simulate the deposit zone, a gated open box was constructed. This report details the procedures taken to develop the model including in-depth discussion on the obstacles, thought process, performance, and evaluation faced when solving this issue.

The chosen solution to combat this issue was a system design denoted the name R2-SPIN2, featuring a rotating bristled blade collection unit, servo actuated containment doors and internalised seesaw, with differential steering through four geared DC motors. To meet the community's requirements, the model needed to fit within a 400 cubic millimetre volume and weigh at most 6 kg. To ensure the system met these requirements each step along the path to the final design was deliberated and tested thoroughly.

This report analytically deconstructs the major systems and subsystems of the R2-SPIN2 unit into detailed components. These components include the collection system, the baseplate/chassis, the drive system, the containment system, and the electrical system. All these systems work in tandem to accomplish the challenge to a high standard.

The final section of this report evaluates the final design's performance when run through the simulation. It breaks down the various tests that were conducted on R2-SPIN2 unit and its components. These tests included stress, driving, flex, and electrical analysis. These tests expressed areas for modification and optimisation of the R2-SPIN2 unit to ensure its full capability and reliability when completing the course.

R2-SPIN2 successfully completed the 2025 Warman Design and Build Competitions track. Its performance on the course showed that in slightly less than 60 seconds all three balls were collected and delivered into the deposit zone whereby the R2-SPIN2 unit signaled its completion of task via an illuminating LED green light. Although the challenge was completed to a high standard, the team believes that further improvements on the system can be made. Examples of such improvements are a more reliable and refined mechanical system to secure the storage door and more improvements to the rotating blade and pully system.

Table of Contents

2. List of Figures

3. List of Tables

4. Problem Definition and Objectives

Introduction:

This report presents the final design developed for the 2025 Warman Design Challenge, in which Earth's student engineers were tasked with addressing an urgent energy crisis on the island of Gondwana. Located on a distant planet at the edge of the Milky Way, Gondwana faces ecological collapse due to the near depletion of its non-renewable energy sources. Without a rapid transition to a sustainable alternative, both plant and animal life will be subjected to extreme seasonal temperatures, placing the entire ecosystem at risk.

Recently, three meteorites rich in rare minerals suitable for sustainable energy generation have landed randomly across the island. However, due to a rapidly increasing atmospheric temperature, the meteorites must be urgently transported into an underground fuel storage bunker. Given the terrain hazards, including a narrow bridge and pivoting seesaw, manual collection has been deemed impossible.

To address this, students were tasked with developing a scaled down autonomous transport system capable of collecting and depositing all three meteorites, represented as a tennis ball, a racquetball, and a table tennis ball, into the designated deposit zone. The system must operate within a tolerance range of 400 cubic mm, weigh no more than 6kg, navigate dynamic and unstable terrain, and complete the task within 120 seconds, starting from a single activation. The resulting prototype demonstrates proof-of-concept functionality, autonomy, and mechanical reliability.

Problem Statement:

To design and construct an autonomous prototype system capable of collecting three spherical meteorites (Tennis ball, racquetball, and table tennis ball) from randomly distributed locations within a defined 3x3 grid, depositing them into a storage bunker (Ball deposit zone) within 120 seconds. The system must be activated by a single initiating action, remain within the 400mmX400mmX400mm volume, and autonomously navigate challenging terrain including narrow paths and a pivoting seesaw platform without human intervention.

Customer Requirements

Below is table 1 is a summary of the customer requirements that must be met by the R2-SPIN2 unit. These requirements were based on the "Warman-design-and-build-rulebook". **Note the extended table is found in Appendix A*.

Table 1 - Summary of the customer requirements (based on the Warman competition rules applicable to our robot).

	Engineering Characteristic	Specification
1	**Ball collection rate**	Collect $(\mathbf{1 \leq balls \leq 3})$ **balls per run (Between 1 and 3 balls)** within **45 seconds**
2	**Positional accuracy of arm (Stepper motor)**	Arm position tolerance within **±1 mm** during operation
3	**Storage capacity**	Store $(\mathbf{1 \leq balls \leq 3})$ **balls** securely during full robot movement
4	**Completion time**	Complete full mission in **≤ 60 seconds with**
5	**Stop distance from edge**	Detect and stop within **≤ 5 mm** of an edge reliably
6	**Battery life / runtime**	Operate at full load for **≥ 120 seconds** on a single charge
7	**Weight of robot**	Total mass **≤ 5.5 kg** (within cube-fit and mobility constraints) but with 500 grams tolerance.
8	**Structural deflection under load**	Frame deflection **≤ 2 mm** when fully loaded and stationary
9	**Number of components accessible for quick maintenance**	≥ 90% of functional modules accessible without disassembly of unrelated parts
10	**Blade spin duration (collection phase)**	Run blade motor for **8 sec total**, with timed with 2 sec tolerance.
11	**Ramp servo actuation angle**	Move from 0° to 60° in **< 0.8 seconds**, accuracy within **±3°**
12	**Door servo response time**	Full open/close cycle in **< 0.5 seconds**, jitter-free
13	**Stepper homing time (limit switch)**	Reach home position in **< 1.5 seconds** consistently
14	**Number of states in state machine**	Use **≤ 12 states**, clearly named and traced in code, can be up to 20 but then may become very complex and hard to code.
15	**Wiring layout clarity**	All wires secured, color-coded, labelled, and routed on designated channels, also protective sleaves.
16	**Ball containment integrity score**	100% containment — **no ball loss** during full run %90 range as 100% target may be unrealistic.
17	**Cube-fit compliance (400mm cubed)**	Entire robot must fit within a **400 mm cube** without compression or force.
18	**Ground clearance consistency**	Maintain **≥ 20 mm clearance** across all terrain transitions (e.g. see saw edges)
19	**End-of-run signal visibility**	LED or signal visible from **≥ 3 meters** at **120° viewing angle. (Green LED for indicating completion)**

Engineering Requirements:

Below is a list of engineering requirements that must be met. These requirements were made based on the course layout for the simulation of the challenge.

- Balls must deposit 30 mm high for the ball to be laced into the ball deposit zone.
- All parts must remain secure once the seesaw has dropped.
- Machine must be able to climb incline of seesaw
    - Must be able to grip seesaw
    - Motors must be strong enough to ascend incline.
    - Must have ground clearance to climb onto see-saw.
- Must be manoeuvrable
- Must be able to control its own speed
- Must be within $400budget
- Must have neat wiring.
- Must withstand see-was drop.

Engineering specifications

Below in table 2 is the GFD Matrix Summary for this design challenge, which entails the engineering specifications for the R2-SPIN2 unit (See Appendix A for full version of the QFD).

Table 2 – QFD Matrix Summary (Full version in Appendix A)

Customer Requirement	Ball Rate	Arm Accuracy	Storage	Completion Time	Stop Dist.	Battery Life	Weight	Deflection	Quick Maintenance Access	Ramp Angle	Door Time	Homing Time
Collects balls accurately	9	9	3	3	3					9	3	
Reliable autonomous navigation	3	9	1	9	9	9	3				3	3
Stores balls securely	1	1	9									3
Robust mechanical structure		3	3			3	3	9	3		3	
Completes task within time limit	3	3		9	3	3	3			3	3	3
Avoids obstacles or table edges		3		3	9							

Accurate ball collection by the system is mostly dependent on quick ball intake, accurate arm motions, and precisely timed ramp and door actuation as shown in the summary QFD, but there are more to consider in the full QFD in the Appendix A. Accurate arm control, quick completion times, quick obstacle recognition, and robust battery life all contribute to dependable autonomous navigation of R2-SPIN2. Robust storage capacity is the main need for secure ball storage; handling precision and placement play a minor role. Precise stop distance of the stepper motor and rotation of the 6V geared DC motors are the essential parts to ensure that all the balls are collected appropriately.

5. Detailed Description

5.1.1 Baseplate, Drive system

The base plate component acts as a mounting point for all subsystems, and is the main body of R2-SPIN2, the wheels mounts, collection sub-system, containment sub-system, electronics mounting subsystem and all electronics systems are mounted upon this component.

The base plate components had to be strong enough to contain the weight of all the systems attached while also holding them sturdily while navigating the courses see-saw. This is easier said than done as the base plate component went through multiple prototypes. All prototypes followed the same principle when it came to mounting systems, and this was using 3mm (m3) standard holes to connect the systems via bolts. Throughout all testing this method proved effective. All prototypes also used a laser cutter to cut the acrylic plates. This production method was quicker and stronger than 3D printing an entire plate. The process and material when laser cutting also benefited the budget as these costs were covered by the university.

The first prototype consisted of two 3mm acrylic sheets connected via two lots of 20mm and 10 mm hex bolt spacers as shown below in figures 1,2, and 3. In this prototype the two connected base plates had a large grid of m3 holes. These holes provided freedom when testing various designs as they provide multiple mounting points and allowed for multiple ideas to be tested. The grid also increased the ease of human use as it could quickly be adapted. The two acrylic sheets had a combined weight of 980g. This was hefty but we thought the cost of the weight was beneficial as thinner, lighter plates would be far too weak.

Figure 1 – First base plate prototype disassembled

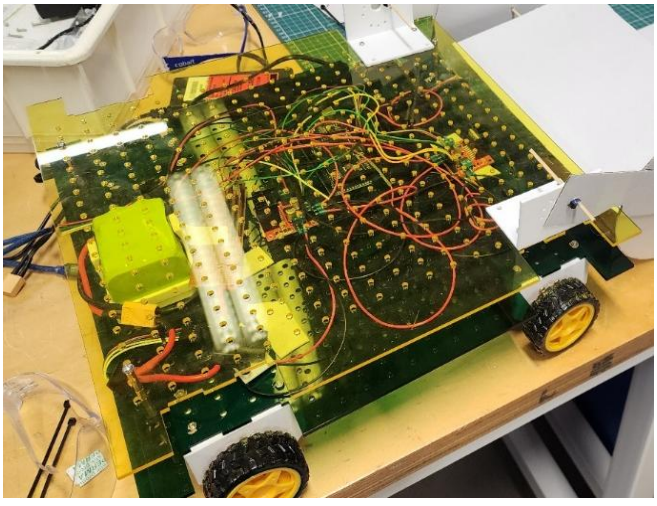

Figure 2 – First base prototype constructed.

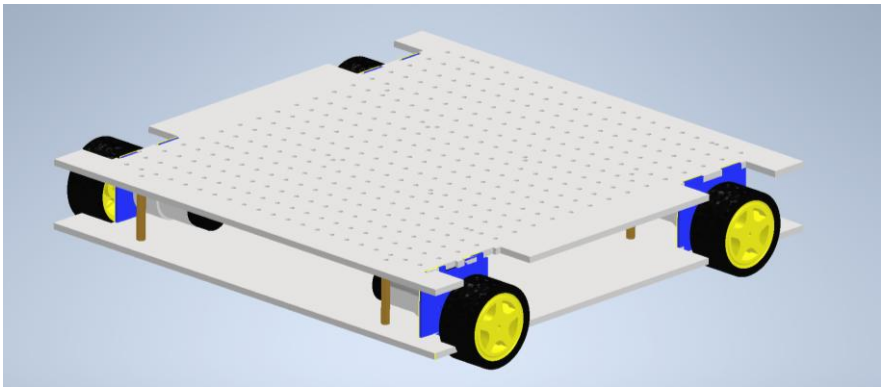

Figure 3 – CAD design for first prototype with base and top plates.

The flaw of this first prototype was in the form of bending and cracking during use showing the flaws of brittle acrylic. When all systems where mounted to this base plate there was large flexing/bending of the bottom plate it was found that the bottom plate flexed by roughly 3mm. During testing more hex bolts were added but this took room away for electronics. The plate did not only bend but after the tipping of the see-saw the acrylic cracked leading to a critical failure, notice the crack on the top left-hand side of the green plate in figure 1. This prototype was also flawed at it was difficult to mount the collection system as the top plate was 60mm above ground level.

The second prototype was created with the goal of creating a lower mounting point for the collection system (see, figure 4 below). This was done by halving the size of the top plate. The idea of the top plate was to add strength to the bottom plate and be a mounting point for the electronics. This prototype still used the m3 grid to mount components. This prototype was quickly dismissed as using one plate with the grid reducing the strength it could not tolerate the stresses it underwent.

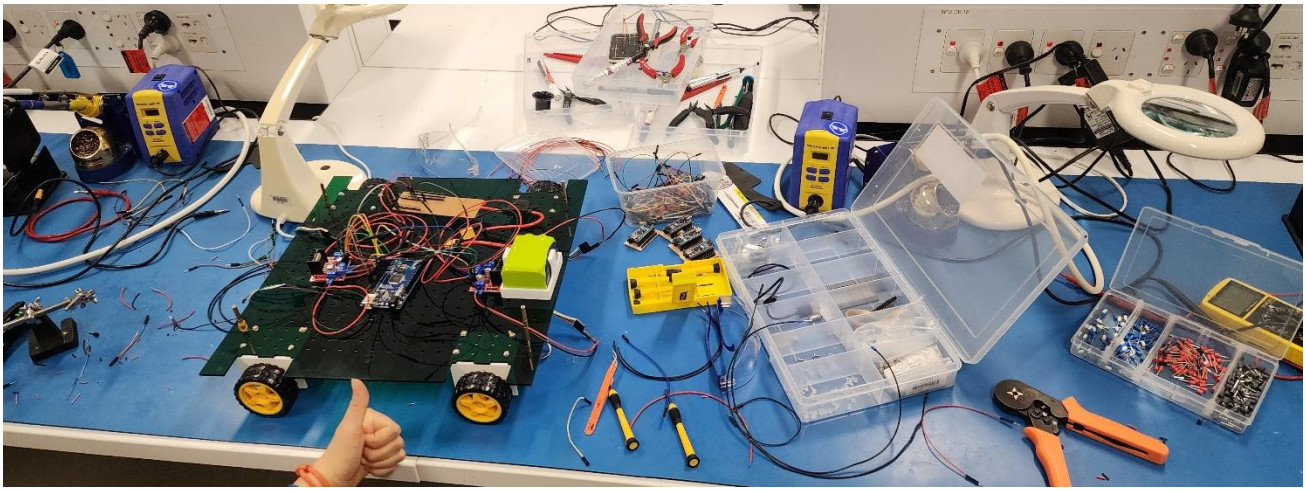

Figure 4 – Prototype 2 with electronics and wheels mounted to singular base plate.

The final prototype was more carefully planned out. It was made using a single 6mm thick sheet of acrylic. This prototype had less holes and was thicker meaning it was much stiffer and less prone to cracking. The final prototype different from the other two as it has slots that allow components from the collection system to be mounted too, giving similar freedom to that of the grid. These slots also provided a means of cable management. Originally, the new base plate had a thickness of 4.5 mm and was using a laser

cutter. The laser cutter settings were set to 3 mm and not 4.5 mm by mistake so to remedy this, two passes were required to produce the base plate. This caused some of the through holes to be slightly misaligned. This plate was deemed undesirable as none of the mounts were correctly aligned, therefore, the final design was mounted on a newly laser cut 6mm acrylic base plate. The layout of the final design base plate can be seen below in figure 5.

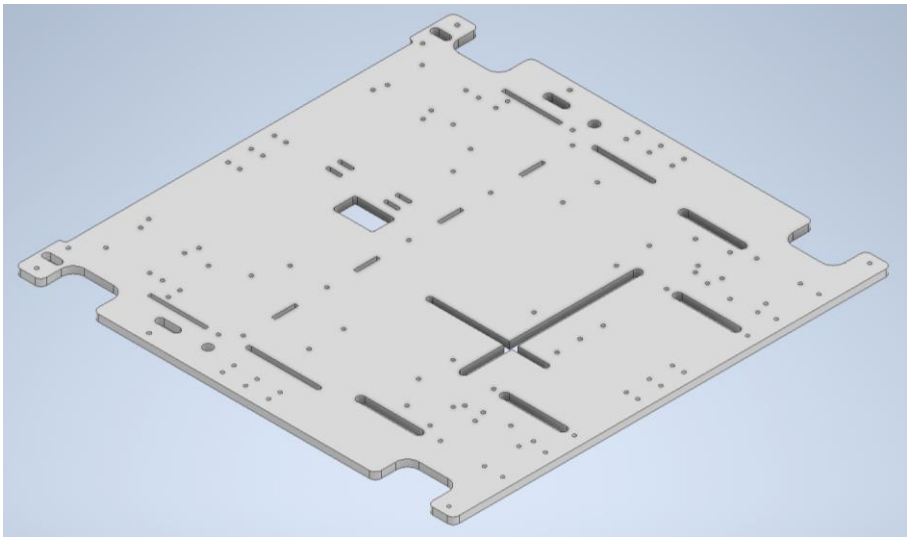

Figure 5 – Final CAD design for base plate of R2-SPIN2 unit.

Mounted to the base plate are the motors. R2-SPIN2 utilises 4 motors and uses differential steering to control its direction. Throughout the prototypes of the drive system, multiple wheels and mounts where used. All the mounts created where 3D printed due to their complex shape (see figure 6). All the motor mounts utilised the screw holes in the motors and a circular section around the axial to hold the motor in place. 4 motors and wheels ensured that R2-SPIN2 could ascend the see-saw with complete control.

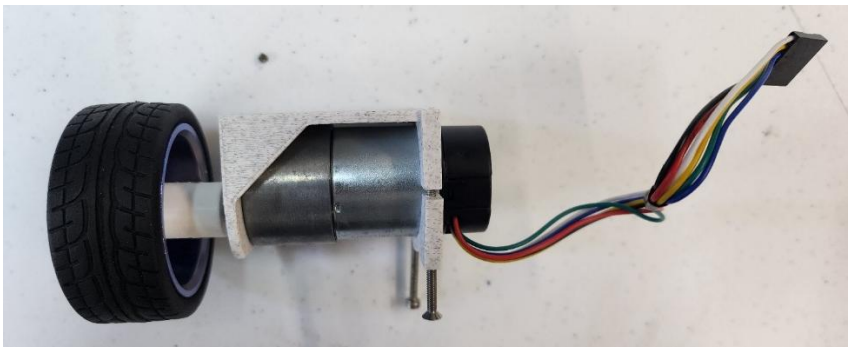

Figure 6 – 3D printed DC motor to wheel mount.

5.1.2 Mounts

The first concept had a square shape, this variant had multiple issues. One such issue was strength; they were easily snapped from compressive force. Another issue was the square shape of the mount, the shape caused issues when trying to climb the see-saw as the mount itself would prevent contact with the seesaw. By the 3rd attempt both issues were solved. Strength was increased by adding supporting structures to either side and chamfering the edges of the mount. A new issue was then encountered which was the motors not sitting flush to the base plate, this caused randomness when steering R2-

SPIN2 as each wheel was at a different angle. This was fixed during testing with a supporting zip-tie. During this same testing phase another issue was encountered which was that the wheels did not sit flush on the motors. This was due to the original wheels not fitting the axel and therefore they were dremeled out. The final version of the motor mount used a new system of a wheel to motor mount connecting with 3 screws shown below in figure 7. This was used in the final design. The final motor mount was smaller and had a part to secure the motor flush to the base.

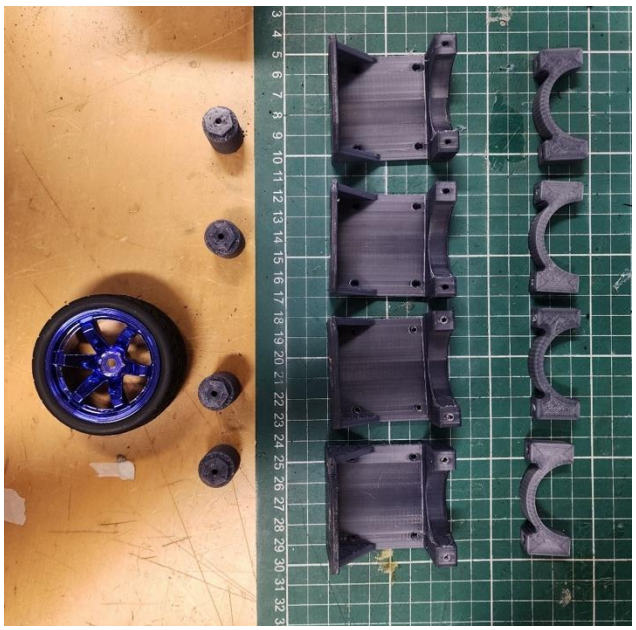

Figure 7 – Final design 3D printed motor mounts with wheel attachments.

5.1.2.1 Electronic Mounting

The electronics mounting system. This system is comprised of 3 levels each level serving a different purpose (See figure 8). The electrical components are mounted to 3D printed plates via m3 threaded bolts. The m3 bolts go through standard mounting holes in the electronics into threaded holes taped via a drill. Using the 3D printer had no impact on the budget. Mounting the electronics at the back of the robot acts as a counterweight to the collection and containment system at the front of the robot. This is beneficial as an even weight distribution will make it easier for the R2-SPIN2 to climb the see-saw.

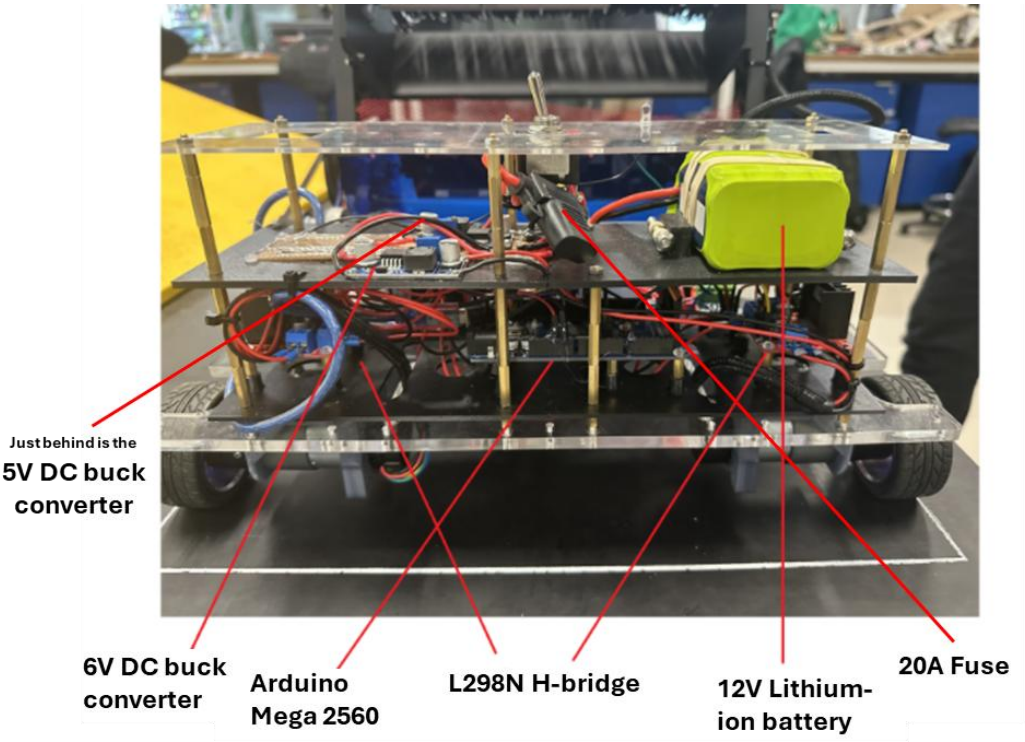

Figure 8 – Side view of electronics mounting system with annotations.

The first level holds the Arduino MEGA and both H-boards. The H-boards are located on the bottom level as they need to be close to the motors making it easier to connect them. The layout can be seen in figure 8 above. The H-board on the left controls the left two motors. The leads from the motors are fed though slots on the base. The Arduino MEGA is mounted between the two H-boards for ease of connection to the H-boards which contain the greatest number of joins.

An elastic band threaded between two slots securely holds the battery in place on the second level. An elastic band is used so the battery can easily be changed for ease of use. The second layer also contains the 5V DC converter and a fuse. These are all mounted on the same layer as they are all directly connected to one another. This layer utilised unused hex bolts from previous prototyping, creating a 60mm gap between the base plate and the first level of the electronic mount, spacing between the first and second levels of electronic mounts was 50mm. The 5V DC converter sits in a rectangular hole in the plate and is secured by hot glue. Finally, the top plate was created for the green LED signal and switch to be located to create a clean and simple user interface shown in figure 9 below.

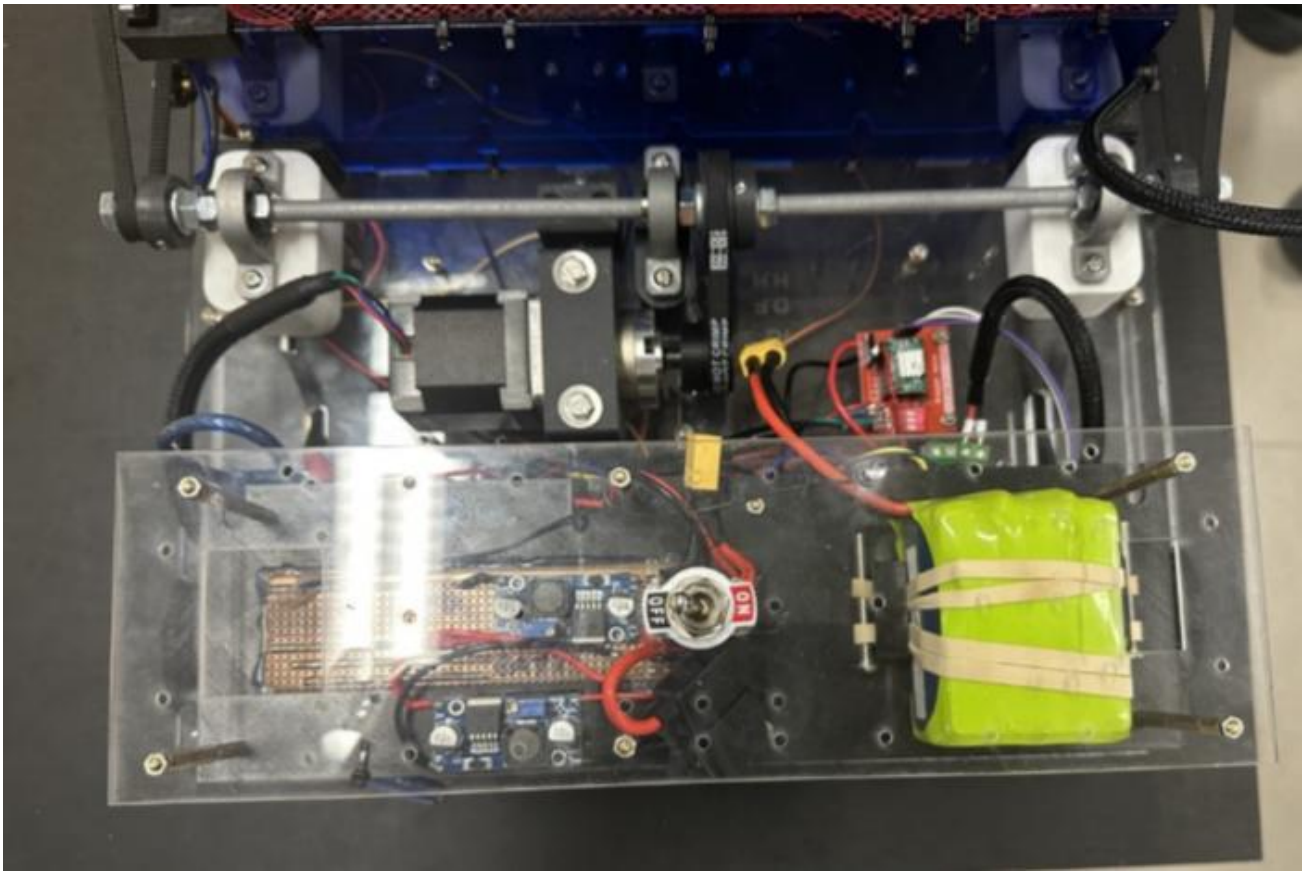

Figure 9 – Electronic mounting system top view.

5.2 Collection System

5.2.1 Description of Final System

R2-SPIN2's collection system consists of a fixed round frame, with a built-in ball ramp, which houses a rotating 'blade' fixed, using nuts and washers, to a threaded rod running horizontally axial through the blade. This blade is rotated by a 6V geared DC 100rpm motor connected to the internal rod via a shaft coupler; to minimise friction, the rod and motor armature are supported by bearings mounted into the round frame on either end (Figure 10).

The frame is attached on either end to two levering arms with a truss design; refer to figure 11 for the following. A timing pulley with 114 teeth is mounted at the base of each arm, these are connected via tensioned GT2 timing belts to 28 toothed timing pulleys which are mounted onto either side of a threaded rod that runs across R2-SPIN2 to power both trussed arms using a Nema 17 stepper motor with gear ratio of 100:1. The stepper motor is mounted semi-centrally on the base plate of the bot and rotates the threaded rod using a 56 toothed pulley on the rod connected via GT2 timing belt to a 28 toothed pulley mounted directly onto the stepper (Figure 12). Using this system of belts and pulleys allowed the collection system to move up when not in use and down to collect the balls, of these components only the rotating blade, round frame and ramp were trialled several times due to complications.

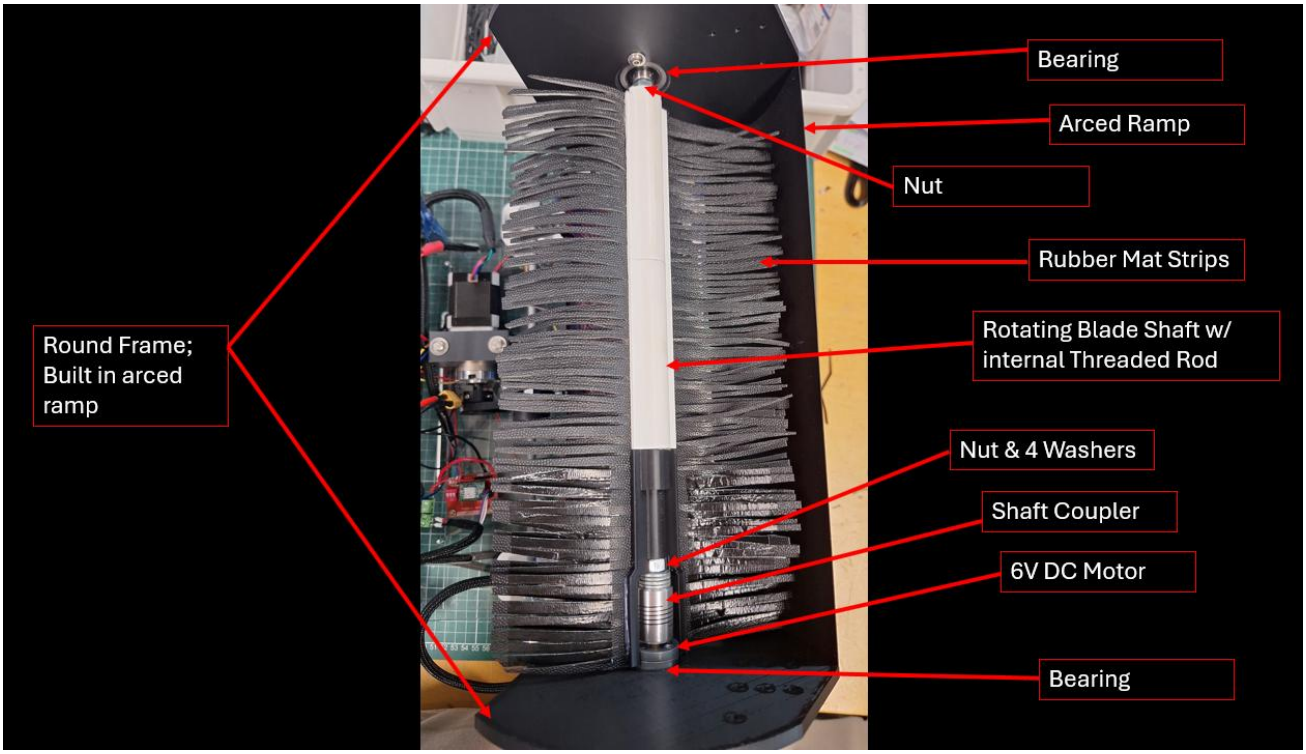

Figure 10 – Analytical deconstruct of collection unit from birds-eye view.

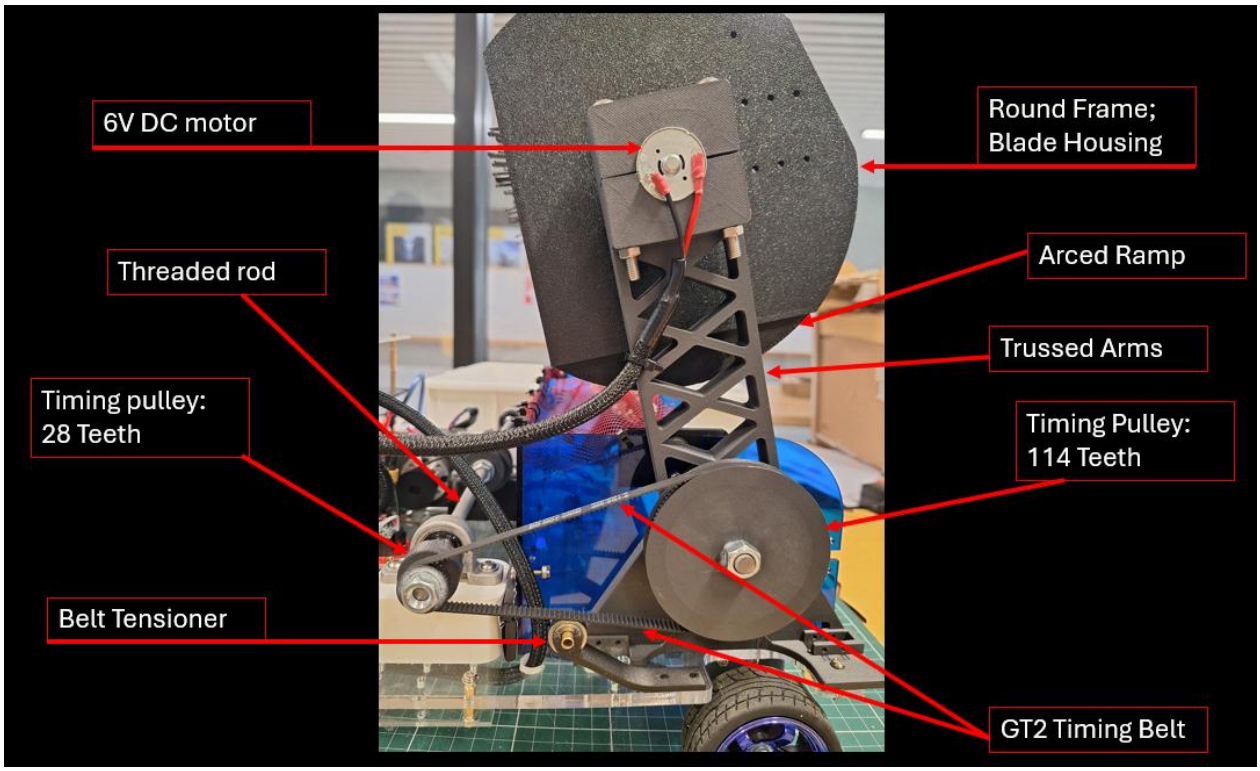

Figure 11 – Sideview of collection system with analytical deconstruct of components.

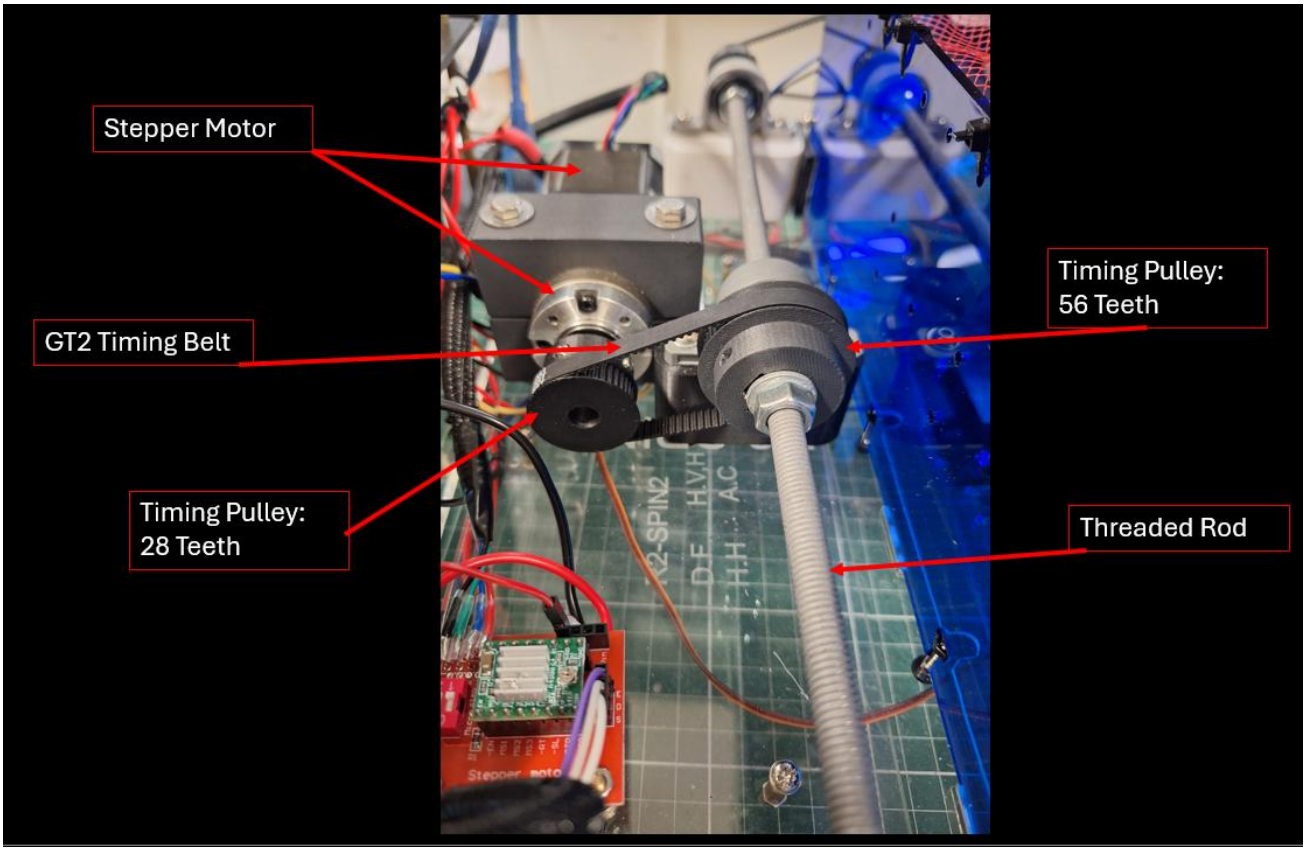

Figure 12 – Collection system's stepper motor connection to the threaded rod.

5.2.2 System Evolution – Frame, Ramp, Blade

Since the round frame and ramp are adjoined by a segment following the arc of the circular walls (Figure 10-11), this 310mm structure was too long to print using the 3D printers provided; it was split into two halves that were connected via the ramp. Initially the ramp was not arced and had a flat edge that acted as a step (Figure 13), hindered the blade's ability to brush the balls up the ramp. This flat edge was replaced with a tapered edge and significantly improved the functionality of the ramp; after testing it several times, the design remained a part of the final collection system (Figure 10-11).

In the design phase, it was thought that the rotating blade could be a solid panel that would strike the balls into the containment unit, after a prototype was constructed and tested it was evident that a rigid blade would cause significant pinching and be prone to snapping if stuck or wedged. To reduce pinching all together, a blade with bristles slotting into receding panelling was trialled; refer to figure 13 for panelling without bristles. Despite utilising bristles and receding the panelling in towards the rotating axis, this blade design had to be scrapped as there was too much friction between the bristles and the ramp as well as severe pinching of the tennis and rubber ball due to the panel. At first this blade was modified heavily to allow the 6V geared DC 100rpm motor to overcome the friction, however the bristles were cut too short to be effective and the severe pinching from the panelling remained, hence a new approach was required.

Figure 13 – Initial concept for rotating blade with lateral inserts for bristles.

The blades panelling was essentially removed entirely, all that remained was a shaft for the threaded rod to be fed through and slits for bristles to be slotted into thus replacing the panelling. Although, before gluing in longer bristles to create a new blade, it was decided that bristles are too springy and would likely cause similar problems to those seen in the previous design. As a result, a slightly smooth rubber mat material was selected for the final design instead of bristles, it was glued into the slits then cut horizontally to a visually preferred length before making numerous vertical cuts in-towards the shaft as seen below (Figure 14). The number of cuts was adjusted several times, making the strips thinner as required; the blade with less cuts had some difficulty picking up the balls as the thicker strips were less pliable, therefore more cuts were made until R2 could collect all three balls at least 90% of the time. What made a big difference with the new rubber mat material was that it could be reformed using a heat gun, it was found that curving the mat strips away from the balls, forming a concave shape (Figure 14 & 17), allowed R2 to collect the balls more easily and reliably. As a result, all three balls could be collected nearly 100% of the time from any configuration on the courses grid layout without labouring the motor or pinching the balls; notably, the more times it was tested the easier it became for R2 to obtain the balls without struggling, this is likely due to the material becoming looser and thereby more forgiving.

Figure 14 – Final design rotating blade unit with rubber mat material bristles inserted accordingly.

5.3 Containment Unit

The containment/storage unit was developed to securely store balls of varying sizes during the course run and release them in a controlled manner at the deposit zone. The unit had to function effectively to overcome obstacles such as the seesaw, under dynamic loading and tipping conditions, and in coordination with the system's collection and actuation subsystems. The final design was developed through multiple iterative prototypes, each addressing observed shortcomings in functional testing.

5.3.1 Containment Unit Structure and Fabrication:

The containment unit was constructed using laser cut 3 mm clear acrylic with final dimensions of 150mmX310mmX100mm. These dimensions were selected to comfortably fit the expected number of balls as well as overshoot the height of each ball whilst also maintaining integrative compatibility with the systems chassis. The original configuration of the containment unit took the form of an open box with equal rectangular side lengths and a longer back wall with a forward-sloping angled base. The angled base was designed to allow the collected balls to naturally roll forward and exit the system once the depositing phase commenced.

Mounting Evolution:

Prototype 1 featured 3D printed L-brackets mounted on the outside of each wall to connect the acrylic walls to the top base plate. In prototype 1 the top base plate was angled forward due to the manipulation of the hex bolt spacings between the top and bottom plates. Whilst structurally sound, the rear L-brackets were 3D printed to match the holes on the base plate which extended them along the back face of the plate which impeded the available space for the components including the stepper motor and electronic mounts.

To address this, shortened L-brackets were introduced with half mounted within the containment unit and half mounted externally. This allowed for the successful installation of other system components; however, the internal brackets introduced obstruction with the balls retainment within the unit, causing frequent disturbances and in some cases unintentional release during testing. To resolve this issue, in the final design all L-brackets were removed and replaced with a male-female interlocking system (see

figure 15). Male inserts extruded from the bottom of each wall of the unit and slotted into complementary female recesses on the base plate. To further secure each wall on the chassis vertical 3mm screw and nut inserts were laser cut (see figure 15). This way a screw could be sent through one of the holes on the underside of the base plate and fastened using a nut. Furthermore, this same method was used horizontally to fasten both side walls to the back wall of the unit. To do this, in the final design it was ensured that each side wall aligned with the thickness of the rear wall.

Further, in the initial design the rectangular configuration of the side walls not suboptimal particularly when lowering the collection system as there was significant interference between the moment arms, step size and the upper frontal corners of the unit. To accommodate for this the upper frontal corners of the side walls were filleted for optimal clearance of the moment arms and step size efficiency (see figure 15). This configuration was extensively tested and verified to prevent interference with the other components of the system as well as the motion of the balls, whilst ensuring structural rigidity throughout the course.

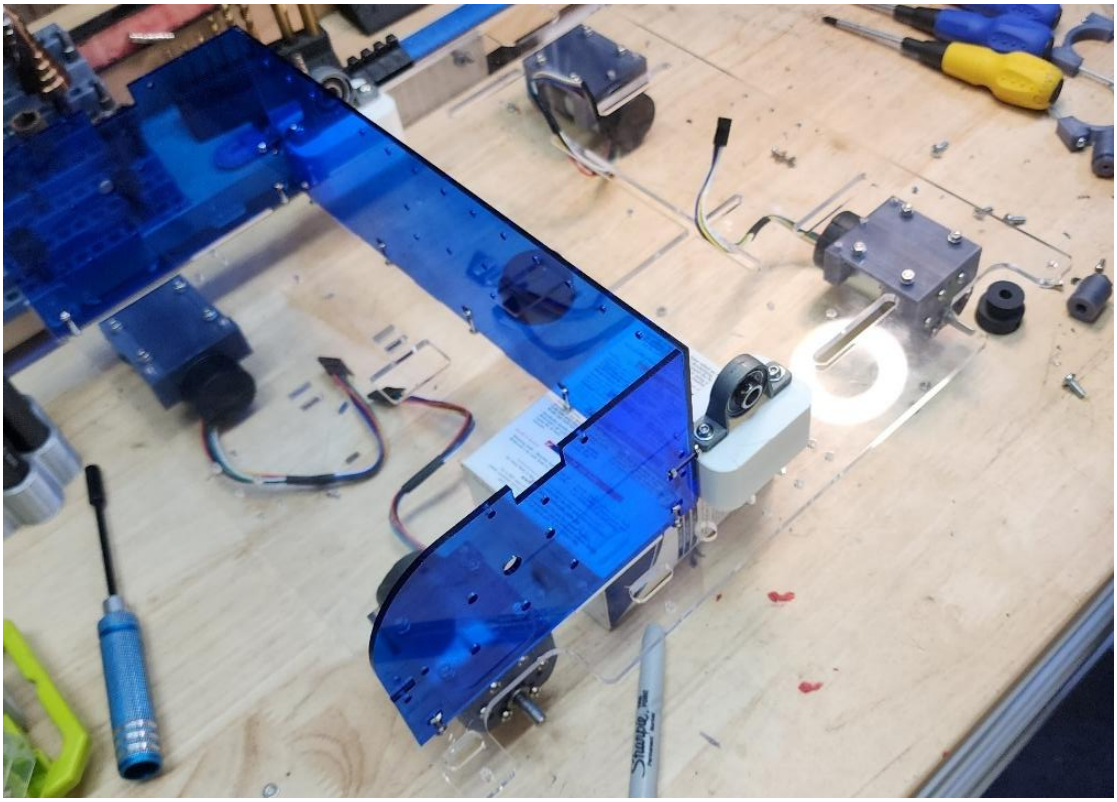

Figure 15 – Containment unit with male-female inserts, horizontal and vertical screw and nut fasteners, and filleted frontal corners.

5.3.2 Angled Base Redesign and Internalised See-Saw Integration:

The initial design incorporated two 3mm acrylic plates which were stacked and offset by hex bolts to create a front-slopped angle. However, this limited the amount of space between the two plates for larger electronics to be mount. Further, through testing it was found that even with a minimal slope the containment unit would not retain the balls at the back of the unit long enough for all other balls to be collected. Often the already collected balls would roll forward and interfere with the collection of others and in some cases cause unintentional release which was undesirable. To resolve this, in the final design

the containment unit was modified to include an internalised see-saw. The internalised seesaw was laser cut out of 2mm acrylic and attached to the chassis via 3D printed attachments. Further, the action of the seesaw was facilitated through the actuation of a micro servo motor connected to a 3D printed fulcrum interface with the seesaw (see figure 16). To optimise the tilting of the internal seesaw, bearings were connected to each side wall and the seesaw. During ball collection, the seesaw remained tilted toward the rear of the unit, ensuring collected balls settled at the back where they remained for the duration of this phase. The seesaw then remained tilted toward the back of the unit throughout transit of the course and when the deposit zone was reached, a servo motor triggered it to tilt forward, enabling controlled forward roll-out of the balls. The FS90 micro servo motor with 180° maximum rotation was mounted to the underside of the chassis via 3D-printed attachments, and its turn dial (servo horn) was fed through a laser cut hole in the base plate to connect with the seesaw and allow for stable pivoting motion (see figure 16). Testing showed that the 3 mm screw and nut fasteners used to secure the unit walls in place would interfere with the full range of motion of the seesaw, especially when tilted toward the back of the system. To resolve this, an outline of each interference zone was dremeled out to ensure full range of motion when tilting the internal seesaw forward and backward. Further testing demonstrated exceptional control over ball positioning and timing of release, significantly improving performance and reliability.

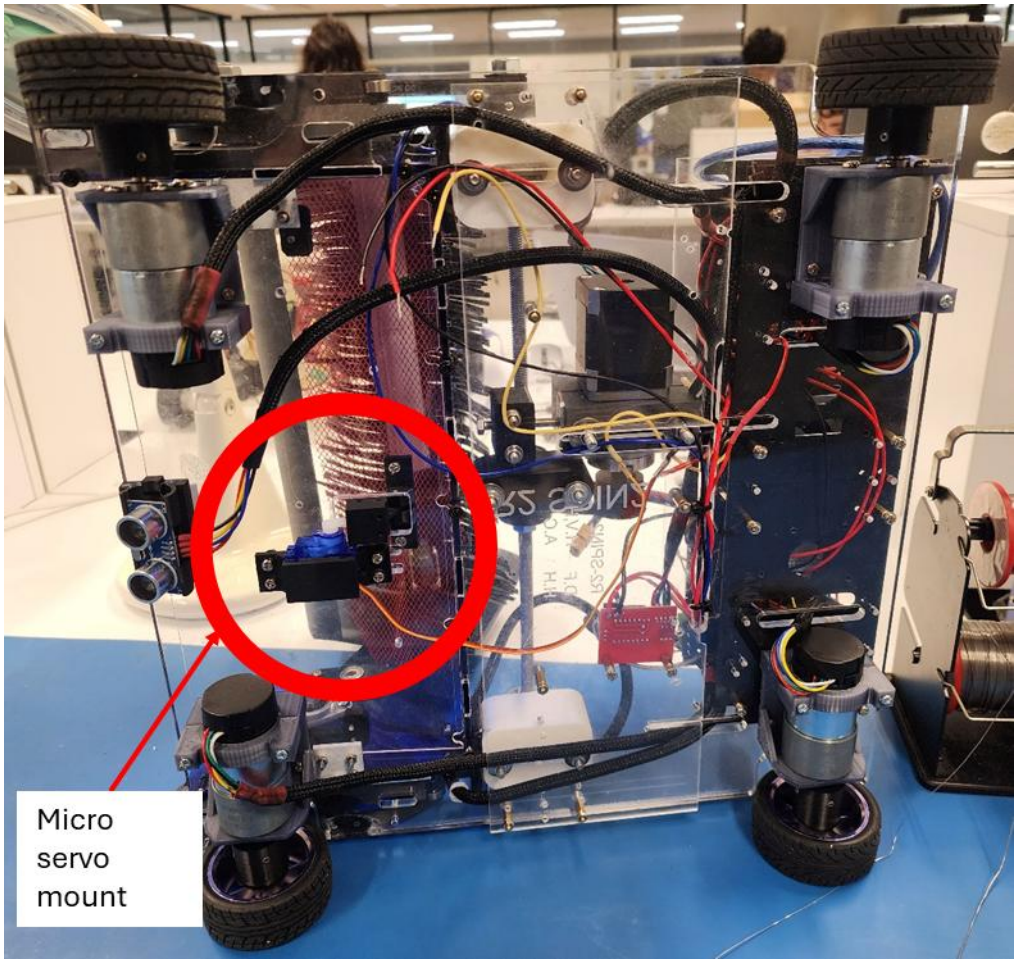

Figure 16 – Containment unit with micro servo, its mount, and connection to internal seesaw as well as dremeled out interference zones.

5.3.3 Servo-Operated Door System:

The containment door was constructed using 2 mm thick plywood and was also servo-actuated to control ball release. Key features included:

- SG90 micro servo motor, capable of 180° rotation, chosen for its lightweight design, cost-effectiveness, and sufficient torque for opening and closing the plywood door.
- Servo mount with embedded female insert for servo horn which sat flush against the side wall via laser cut holes.
- Bearing mount fit for 6mm bearing on left side wall connecting 25 mm 3D printed rod to actuated door. Ensuring smooth rotation and door alignment over the span of the 30 cm width.

This dual-mount system enabled free rotation of the door while minimising load on the servo, thus preserving torque and reducing power draw. Testing showed successful raising and lowering of the door for easy collection and depositing of the balls. However, it was observed that through transit especially during the transit over the course seesaw, once tipped the balls would bounce or roll forward and knock the servo door open leading to unintentional release. This was found to be due to the low ground clearance of the door to the internal seesaw's base which would flex open under load leading to early releases. To resolve this masking tape was used to construct a 'tail'. The masking tape 'tail' was constructed by adding tape at the bottom of the door on both sides, which in turn bridged the gap between the bottom of the door and the seesaw base. This masking tape tail was then further extended by the addition of another row of tape, this ensured that when the balls would knock the door, they would roll on top of the masking tape tail which would then impede the door from opening. This simple yet effective solution maintained full control over the deposit timing and was robust across multiple testing scenarios.

5.3.4 Containment During Dynamic Tipping:

During testing on the course's seesaw, it was observed that on the incline the balls would climb the rear and side walls and escape the containment unit. Further, on the decline, due to tipping the balls would bounce out of the containment unit. To resolve this 3 mm laser cut holes were added along the top of the rear and side walls of the containment unit. Using these holes, a mesh net was secured using zip ties to form a flexible yet effective containment cover (see figure 17). This prevented all observed ejections during further testing on the incline and decline of the course's seesaw as well as all other components of the course.

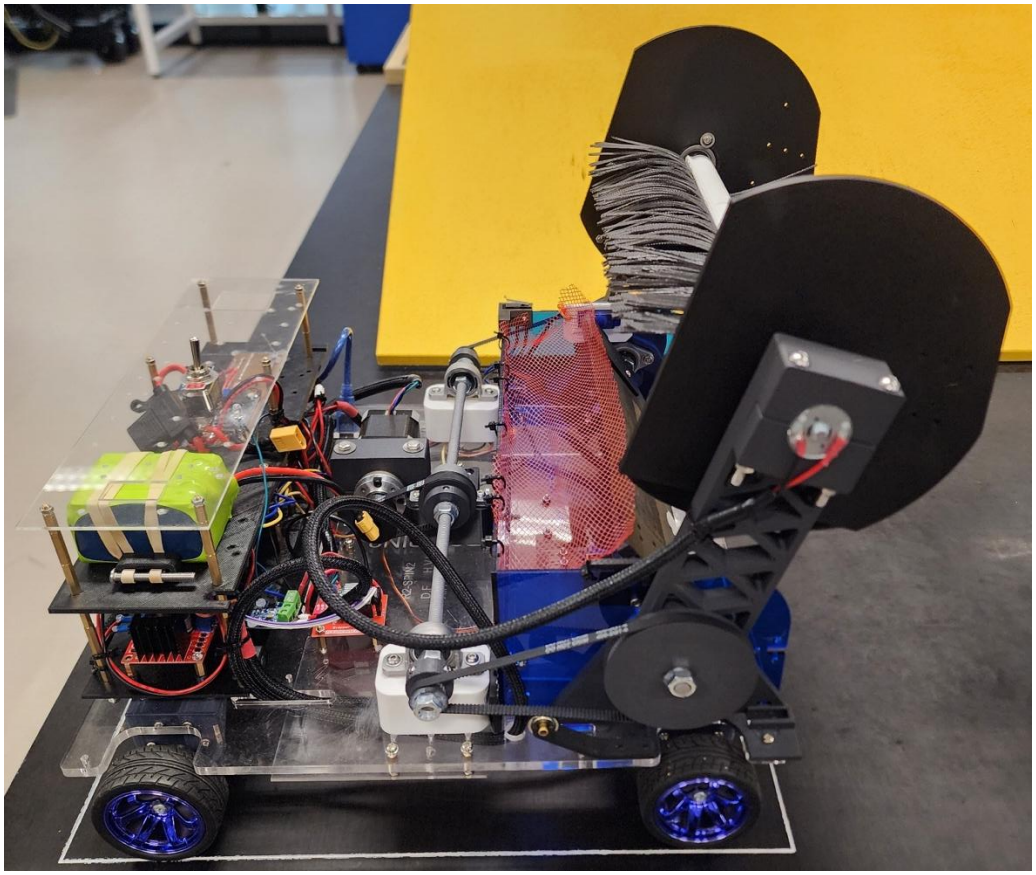

Figure 17 – Containment unit with mesh net cover

5.4 Electronics & Programming

This section outlines the electrical systems and programming logic that enable the robot's autonomous operation. Each component was selected based on compatibility with the Arduino Mega 2560 and its ability to meet the mechanical and control requirements of the Warman obstacle course.

5.4.**1 Overview of Electronic Architecture**

An Arduino Mega 2560 serves as the foundation for the robot's electronics system, which manages all the actuators and sensors. A 12V battery pack made up of six 18650 Lithium-Ion batteries supplies power *(Provided by Flinders)*. Two DC buck converters are used to distribute and control the 12V supply to the necessary voltages of 5V and 6V. The 12V directly powers the 4 DC motors driving the wheels and powers the stepper motor. The primary power line has a 20A inline fuse to protect against any overcurrent. This is shown in the figure 18 below.

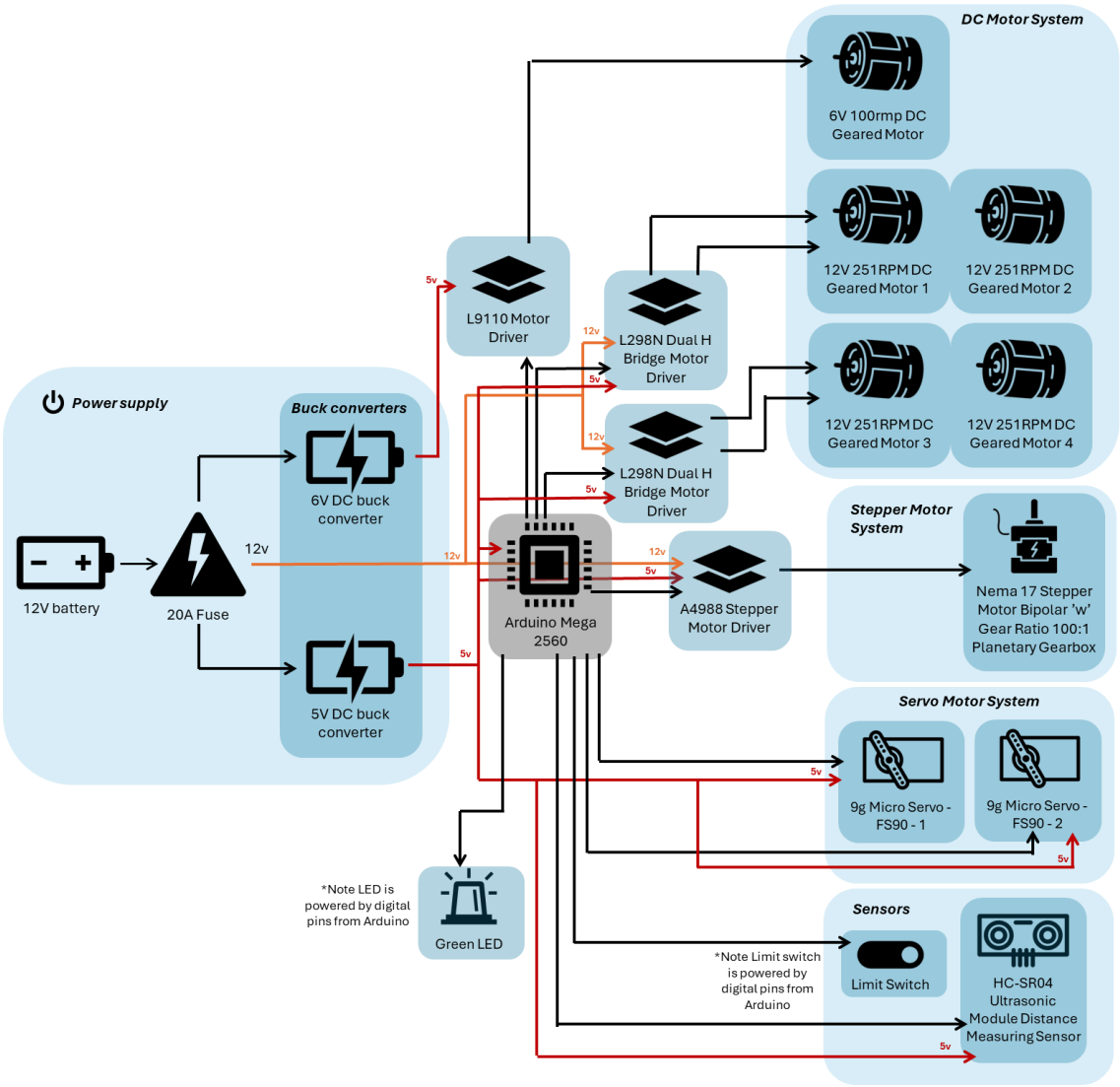

Figure 18 – System Power and Signal Diagram of the electronic systems

5.4.2 Power Distribution and Protection

- **Main Battery:** 6x 18650 cells (12V output)
- **Buck Converter 1:** Steps down 12V to 5V for Arduino Mega, servos, and logic
- **Buck Converter 2:** Steps down 12V to 6V for the 100RPM DC geared motor used in the blade mechanism
- **20A Inline Fuse:** Protects all electronics from overcurrent damage

5.4.3 Motor Control System

Four 12V 251RPM Metal DC Geared Motors with encoders, managed by two L298N motor drivers connected to the Arduino Mega, provide the robot's motion. Two motors are controlled by each L298N. Velocity and directional feedback are provided by encoder signals, which are coupled to interrupt-capable pins, but encoder integration was tedious and had issues with the code incorporation, therefore, the encoders were not utilised.

5.4.3.1 Drive System – DC Motors and L298N Drivers

The robot's mobility is enabled by four 12V 251RPM Metal DC Geared Motors with Encoders, controlled by two L298N motor drivers. Each L298N controls two motors. The wiring is seen below in figure 19.

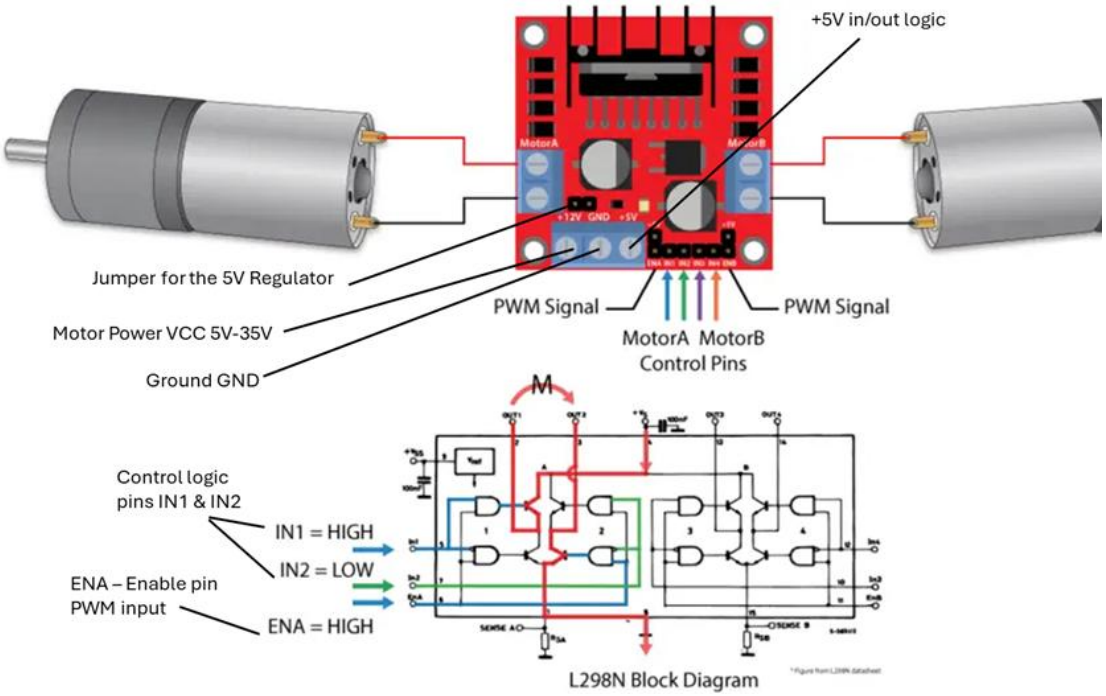

Figure 19 - L298N Wiring with 12V Motors (Dejan, 2019).

5.4.3.2 Arm System – Stepper Motor and A4988 Driver

The robotic arm uses a stepper motor with an A4988 driver, which enables precise control over angular position. The stepper is responsible for raising/lowering the arm via a system of timing pulleys which can be seen in Appendix D Figure D.5**.** A limit switch is used for homing the stepper motor. The position is zeroed when the switch is triggered, and motion control continues via the AccelStepper library in the Arduino IDE (Programming software).

5.4.3.3 Blade and Door Mechanisms – DC Motor with L9110S and Servos

A separate 6V 100RPM geared DC motor, controlled via a L9110S H-Bridge driver, spins the robot's collection blade. This driver receives PWM input for speed control and a second pin for direction reversal and is coded to move when the robot moves forward to collect the balls.

Additionally, two micro servo motors are used to actuate:

- A door for containment of the balls once collected.
- A ramp to tilt downwards when the balls are to be deposited.

Both are controlled using the Servo library and triggered during predefined states in the robot's state machine within the code.

5.4.4 **Edge Detection – Ultrasonic Sensor**

The robot's front is equipped with an HC-SR04 Ultrasonic Distance Sensor, which detects drops and table edges (figure 20). It works by sending out an ultrasonic pulse and timing how long it takes for it to return. The robot stops and switches to a safe mode if the recorded distance falls below a threshold, which denotes an edge.

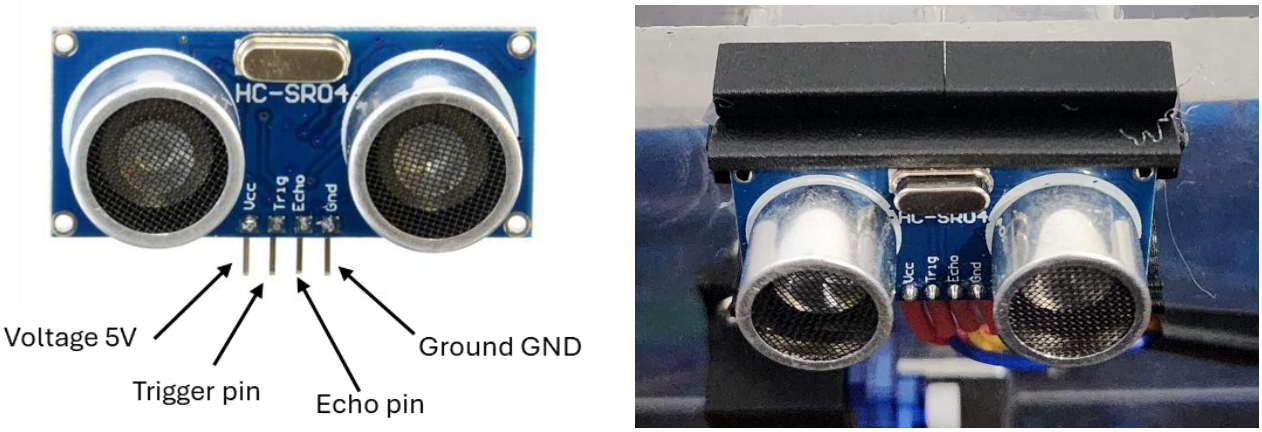

Figure 20 **-** HC-SR04 Ultrasonic Sensor (LEFT) Mounted (RIGHT) for Edge Detection on Front of Robot

Trigger Pin: Connected to digital pin 26 on Arduino Mega.

Echo Pin: Connected to digital pin 24.

Threshold Distance: 60cm used to determine an edge off the table.

```
85  //Ultrasonic
86  const int trigPin = 26;
87  const int echoPin = 24;
88  float distance = 0.0;
89  unsigned long ultrasonicCheckInterval = 100;  // check every 100 ms
90  unsigned long lastUltrasonicCheck = 0;
91  const int threshold = 60; // cm
92  bool edgeDetected = false;
```

Figure 21 – Summary of Arduino code detailing threshold distance and other tolerances for Ultrasonic Sensor.

5.4.**5 Software and Logic Architecture**

All components are managed by a state machine running on the Arduino Mega. The program structure:

```
25  enum State {
26    HOMING,
27    MOVE_TO_POSITION,
28    RUN_MOTORS_FORWARD,
29    SLOW_DOWN_FOR_COLLECTION,
30    WAIT_FOR_COLLECTION,
31    RETURN_HOME,
32    FINAL_STOP
33  };
```

Figure 22 – Summary of Arduino code detailing the program structure of the systems run through of the course.

Those states have sub states that are linked with Boolean logic, which executes complex robot behaviours based on a series of logical conditions and sensor feedback.

1. **Homing –** The stepper motor homes the arm using a limit switch to establish a known starting position.
2. **Move to Position –** The stepper motor moves the arm a set distance forward (49,100 steps) toward the collection zone.

```
206         stepper.moveTo(40100);
```

Figure 23 – Arduino code detailing step size used by stepper motor to lower the collection arms.

3. **Run Motors Forward –** The four DC drive motors activate and move the robot forward and stop whether the ultrasonic sensor detect the edge or a fail-safe timing-based variable of 2 seconds of the ultrasonic sensor gives back false readings as ultrasonic sensors tend to fail.

```
235     case RUN_MOTORS_FORWARD:
236       if (millis() - lastUltrasonicCheck >= ultrasonicCheckInterval) {
237         lastUltrasonicCheck = millis();
238         distance = readUltrasonicDistance();
239         Serial.print("Distance: ");
240         Serial.println(distance);
241         if (distance > threshold && distance < 300 && distance > 0) {
242           edgeDetected = true;
243         }
244       }
245       if (edgeDetected || millis() - actionStartTime >= 2000) {
246         stopAllMotors();
247         digitalWrite(BLADE_EN_PIN, HIGH);
248         analogWrite(BLADE_PWM_PIN, pwmValue_2);
249         currentState = WAIT_FOR_COLLECTION;
250         actionStartTime = millis();
251         edgeDetected = false;
252       }
253       break;
```

Figure 24 – Arduino code, case based, used to program the R2-SPIN2 unit to move forwards.

4. **Slow Down for Collection –** The robot slows its movement to prepare for object collection.
5. **Wait for Collection –** During this timed phase, the blade motor spins for 8 seconds to ensure all balls are collected in the millis() – actionStartTIme >= 8000. Then performs the next tasks such as closing the servo door and turning off the blade then moves to next state after defining the speed percentage in PWM signals that the DC wheel motors will get.

```
255     case WAIT_FOR_COLLECTION:
256       if (millis() - actionStartTime >= 8000) {
257         closeDoor();
258         digitalWrite(BLADE_EN_PIN, LOW);
259         analogWrite(BLADE_PWM_PIN, 0);
260         stepper.moveTo(0);
261         actionStartTime = millis();
262         dcStartedBack = false;
263         speedPercent = 75;
264         pwmValue = map(speedPercent, 0, 100, 0, 255);
265         currentState = RETURN_HOME;
266       }
267       break;
```

Figure 25 – Arduino code, case based, used to wait for the collection system to collect the balls.

6. **Return Home –** The stepper motor brings the arm back to the home position while drive motors perform additional movements (e.g., reversing, turning, traversing the see saw and depositing the balls).
7. **Final Stop –** All motors stop, and the robot enters an idle or reset state where a green LED turns on signalling task done.

5.4.6 Justification of Design Decisions

- **Arduino Mega** was selected for its high I/O pin count and compatibility with interrupts and multiple PWM channels.
- **L298N drivers** are robust and readily compatible with 12V motors control.
- **L9110S driver** was used for the smaller 6V motor, as the larger L298N with its heatsink was unnecessary for the lower power requirement.
- **Buck converters** allow efficient power splitting without overheating or overloading logic-level devices.
- **A4988 stepper** control provides accurate arm positioning necessary for successful object retrieval.
- **Fuse protection 20A** ensures robustness and user safety.

5.4.7 Interaction Between Software and Hardware

All hardware components are controlled by the Arduino software.

- The **limit switch** defines a mechanical zero for the stepper arm.
- The **servos** are activated by flags within the collection state.
- The **blade motor** sequence is implemented using non-blocking timers within the loop.
- Also, the 4 DC motors for the wheels uses the millis() function for non-blocking.

6 Device Evaluation

6.1 Description of Test Plans

R2-SPIN2 successfully completed the 2025 Warman Design and Build Competitions track, in slightly less than 60 seconds, with all three balls collected and delivered into the deposit zone before R2 indicated the task was complete by illuminating a green LED. To achieve this, R2 was subjected to various tests throughout the design, build, and programming phases.

Early on during the design and build phase the first round of 3D printed parts and acrylic components required a plethora of testing including: physically ‘stress’ testing the first un-filleted motor mounts which snapped in hand when squeezed; flexing the first two plates of laser cut yellow and green acrylic to grasp the degree of bending that may occur under the weight of heavy components and eventually having to replace the two acrylic plates as one cracked during a driving test as there was little structural soundness given the weight and all the holes in the acrylic. Notably, each time a new acrylic base plate was machined the motor mounts, motors and wheels were attached to confirm whether R2 would fit within the 400x400 starting box dimensions; this was done to ensure eligibility in the competition requirements before moving onto attaching all necessary components.

Within the build phase, several components required fitments and adjustments to function optimally, take the first set of wheels for example, where each wheel was dremeled to fit onto the motor armatures and had to be replaced by different wheels and 3D printed wheel mounts since the first set were

incredibly wonky and caused R2 to drive poorly. This fitment process was required for other 3D printed components and systems such as the collection system and component mountings.

The building and initial programming phase consisted of movement tests ranging from moving only the motors while suspending R2 to traversing the course and ensuring R2 had clearance to ascend the seesaw. It also included mounting almost all the electronics and ball collection and containment systems in such a way that weight was distributed as uniformly as possible to ensure R2 would be able to traverse the seesaw without sliding or labouring.

6.2 Final System Testing

This phase of tests proved to be complex as innumerable adjustments to all sections had to be made quite frequently to ensure progress was made. Difficulties regarding the ball collection system and traversing the course were most prominent and time consuming; each ball configuration needed to be tested to ensure pinching was minimised and different blade speeds were trialed to determine the optimal speed for collection on the track.

Whereas movements such as turning needed to be repeated on the floor and on the track to ensure R2 could make the correct turn and would not break from driving off the track table whilst adjusting its turning circle. Similarly, traversing the ramp required several practices to determine the correct speed R2 should travel at to prevent the seesaw from slamming onto the table causing damage to R2 or knocking the balls out of containment. These adjustments to the program were made to improve individual movements and thereby increase efficiency overall, doing so allowed each movement required to traverse the course to be isolated, *perfected,* and then reorganised to operate sequentially as a set of movements.

6.3 Test Plans and Methodology

Testing was structured into three main phases as described below:

1. **Subsystem Verification:**
   - Mechanisms for Collection and Deposit was done by consistent pickup and deposit success rates that were guaranteed through repeated experiments and slight adjustments with the blade design.
   - Navigation and Obstacle Avoidance done with ultrasonic sensor and used to evaluate the autonomous navigation system. The response accuracy of ultrasonic sensors and encoders was checked and calibrated, yet encoders were tedious therefore the ultrasonic sensor proved to work the best under the time crunch and ease of coding.
   - Power Consumption was done to ensure effective power utilisation within battery specifications, current draw was tracked using a digital Multimeter throughout operational modes, refer to Appendix C.

2. **integrated System Testing:**
   - Within the 400mm^3 size restriction, the robot was put through complete cycles that included collecting the balls, transporting, and depositing balls. Each phase's timing was noted to evaluate adherence to deadlines, there were troubles with the ball transporting as the balls kept falling out, therefore a quick fix with masking tape on the door fixed the problem, refer to Appendix Figure D.1.

- By positioning the robot close to table edges and confirming motor shutdown procedures to avoid falls, edge detection using the HC-SR04 ultrasonic sensor was confirmed.
- To avoid component damage or harm, safety features such as mechanical shielding and braided wrap cable management sleeves which are fireproof was utilised.

3. **Performance Benchmarking:**

- From the Warman competition rules the DISTURB, MOVE, DEPOSIT, END, and RUNTIME scores were among the competition criteria used to rate the robot's mission performance (Engineers Australia, 2025). Repeatability and resilience were evaluated throughout several runs and summarised results in the results table below.

Below in table 3 is a summary of the test plans with results measured against the key engineering specifications.

Table 3 – Summary of Test Plan Results Against Key Engineering Specifications.

Specification	Target	Result	Pass/Fail	Notes
Ball collection accuracy	≥ 70% success rate	76% success rate	Pass	Minor failures due to ball positioning
Autonomous navigation reliability	No collisions or edge falls	No collisions; 1 edge event in 10 trials	Pass	Ultrasonic sensor required recalibration
Task completion time	≤ 120 seconds	Avg. 77 seconds	Pass	Faster runs observed after optimization
System size	≤ 400mm cube	395mm x 395mm x 360mm	Pass	Slight margin for internal components
Power consumption	≤ 12V, 5A max	Avg. 2A peak	Pass	Efficient power management confirmed
Safety features	No exposed hazards	Fully enclosed	Pass	Protective casing installed *(Braided Wrap Cable Management Sleeve that are fireproof)*
Maintenance accessibility	Easy access to parts	Moderate; improved labelling needed for what pins in microcontroller.	Partial	Some components tightly packed

*Note – Power consumption, see Appendix C, showed the max number of Amps being 5A that the system drains when all systems are on and moving at the same time. Also, all systems won't be on at the same time, therefore the system cannot use 5A at a single moment as each state runs a component at a different time.

6.4 Potential Improvements

Due to time constraints and running last minute tests, it became evident towards the end that R2's trap door constraining the balls and collection systems spinning blade could be improved. An issue that was seen in late tests was R2 dropping the balls once the seesaw tipped, to aid the situation in the moment, tape was added to the base of the trap door to minimise the gap between the door and the containment units base, thus preventing the unintentional release of the balls from the system. This temporary fix could be replaced with a flat, thin almost fabric-like material to act as a door sweep; it may be attached to the wooden door via glue or staples.

Although the rotating blade performed well when collecting balls from numerous configurations, it still struggles occasionally with minor pinching causing the motor to labour, inching the balls up the ramp into the containment unit eventually. Improving the efficiency of this system could be achieved by testing smoother brush-like materials and increasing the radius of the collection frame and thereby increasing the blade radius to further minimise pinching of larger balls. It is worth nothing that given R2-SPIN2's performance, it may be deemed a valuable contender in the 2025 Warman Design Competition and for the scope of this competition R2's design and build are considered very reasonable.

Conclusion

R2-SPIN2 can be deemed a successful design for the Warman design challenge 2025, as it met all the requirements and engineering specifications, R2: completed the challenge withing 120 seconds; fits within the 400mmX400mmX400mm size constraints and weighing $\approx 5.8kg$; traversed the see-saw with complete control; and collected and delivered all three balls. A high standard of completion was achieved given each system performed synergistically, this is evidenced by: the collection systems innovative rotating rubber-stripped blade which successfully and consistently collected the balls whilst connected to two moving arms when in an upright position allowed for R2 to fit within the size constrains; the ball containment system's internal see-saw and adapted door which constrained the balls, thus preventing ball loss whilst R2 was in motion; R2's ability to navigate the course via the drive unit consisting of four motors and four gripped rubber wheels that provided ample traction on the track; and the vital and elegantly executed electrical and electronic system powering and controlling R2's every action. Given the success of R2-SPIN2, the team believes a full sized and slightly adapted version of R2 could produce similar if not better results on Gondwana through transporting meteorites into the underground fuel bunker effectively and efficiently.

References

- Dejan (2019). *Arduino DC Motor Control Tutorial - L298N | PWM | H-Bridge - HowToMechatronics*. [online] HowToMechatronics. Available at: https://howtomechatronics.com/tutorials/arduino/arduino-dc-motor-control-tutorial-l298n-pwm-h-bridge/.
- Engineers Australia (2025). *38th Warman Design and Build Competition, Competition rules*. [online] Available at: https://warmandesignandbuild.org.au/wp-content/uploads/2025/02/2025-Warman-Design-and-Build-Competition-Rules-FINAL.pdf.

Appendices

Appendix A -Engineering Specification Support Material

Includes the full QFD matrix, any extended customer requirement tables, engineering target breakdowns, and risk assessment.
"Full QFD matrix referenced in Section 4.5"

Table 4 – *Risk Assessment Table for R2-Spin2*

#	Risk Description	Potential Impact	Likelihood (L/M/H)	Severity (L/M/H)	Risk Level (L/M/H)	Mitigation Strategies	Contingency Plan	Responsible
1	Overcurrent causing damage to electronics	Electronics failure, fire hazard	Medium	High	High	Use 20A inline fuse; choose appropriate wire gauges; current monitoring during testing with multimeter	Replace fuse; inspect wiring; shutdown system immediately	Electrical Engineer
2	Battery overheating or thermal runaway	Fire, damage to battery pack	Medium	High	High	Use quality 18650 cells; implement thermal monitoring; avoid overcharging/discharging; provide ventilation.	Emergency power shutdown; fire suppression blanket	Electrical Engineer
3	Incorrect voltage output from DC buck converters	Damage to motors, microcontroller malfunction	Low	High	Medium	Test converters under load; use voltage regulators with overvoltage protection.	Disconnect power; replace faulty converter	Electrical Engineer
4	Arduino Mega microcontroller failure	Robot stops functioning	Low	High	Medium	Use proper programming and watchdog timers; ensure stable power supply.	Reset MCU; reload code; have spare microcontroller	Electrical Engineer
5	Faulty motor driver or DC motors overheating	Motor failure or fire hazard	Medium	Medium	Medium	Use motors within rated specs; monitor current and temperature; adequate heat sinking.	Replace motors; cool down periods; redesign if persistent	Mechanical Engineer
6	Wiring shorts or loose connections	System resets, intermittent faults, fire risk	Medium	High	High	Secure and organize wiring; use protective sleeves; perform continuity tests.	Inspect and repair wiring; isolate faulty section	Electrical Engineer
7	Limit switch failure or misalignment	Incorrect homing behaviour; mechanical collision	Low	Medium	Low	Regular mechanical inspection; proper mounting and alignment checks.	Manual override; recalibrate limit switches	Mechanical Engineer
8	Sensor malfunction (e.g., ultrasonic sensor failure)	Robot fails to detect edges or obstacles	Medium	Medium	Medium	Implement sensor self-tests; use sensor fusion if possible	Use fallback behavior; switch to alternative sensors	Systems Engineer

9	*Coding bugs or unexpected state machine behaviour*	*Unpredictable robot behaviour, mission failure*	*Medium*	*High*	*High*	*Code reviews; unit and integration testing; simulation before deployment*	*Emergency stop command; manual control override*	*Software Engineer*
10	*Battery pack voltage dropping below minimum threshold*	*Robot shutdown or erratic behaviour*	*Medium*	*High*	*High*	*Use battery monitoring system (BMS); low voltage cutoff protection*	*Safe shutdown sequence; recharge or swap battery pack*	*Battery System Specialist*
11	*Mechanical structural failure under load*	*Robot damage, loss of functionality*	*Low*	*High*	*Medium*	*Use adequate materials; structural analysis; physical stress testing*	*Repair or replace damaged parts* ***(Which our team had to do)***	*Mechanical Engineer*
12	*Electromagnetic interference, causing signal issues*	*Ultrasonic Sensor/motor control errors*	*Low*	*Medium*	*Low*	*Use shielded cables; proper grounding and layout; filtering in hardware/software, as* ***Ultrasonic can have issues.***	*Reset system; relocate antenna or sensor; shield components*	*Electrical Engineer*

Figure A.1: Full QFD

Table A.2: *Extended customer requirement table*

	Engineering Characteristic	Specification
1	**Ball collection rate**	Collect $(\mathbf{1} \leq \boldsymbol{balls} \leq \mathbf{3})$ **balls per run (Between 1 and 3 balls)** within **45 seconds**
2	**Positional accuracy of arm (Stepper motor)**	Arm position tolerance within **±1 mm** during operation
3	**Storage capacity**	Store $(\mathbf{1} \leq \boldsymbol{balls} \leq \mathbf{3})$ **balls** securely during full robot movement
4	**Completion time**	Complete full mission in **≤ 60 seconds with**
5	**Stop distance from edge**	Detect and stop within **≤ 5 mm** of an edge reliably
6	**Battery life / runtime**	Operate at full load for **≥ 120 seconds** on a single charge
7	**Weight of robot**	Total mass **≤ 5.5 kg** (within cube-fit and mobility constraints) but with 500 grams tolerance.
8	**Structural deflection under load**	Frame deflection **≤ 2 mm** when fully loaded and stationary
9	**Number of components accessible for quick maintenance**	≥ 90% of functional modules accessible without disassembly of unrelated parts
10	**Blade spin duration (collection phase)**	Run blade motor for **8 sec total**, with timed with 2 sec tolerance.
11	**Ramp servo actuation angle**	Move from 0° to 60° in **< 0.8 seconds**, accuracy within **±3°**
12	**Door servo response time**	Full open/close cycle in **< 0.5 seconds**, jitter-free
13	**Stepper homing time (limit switch)**	Reach home position in **< 1.5 seconds** consistently
14	**Number of states in state machine**	Use **≤ 12 states**, clearly named and traced in code, can be up to 20 but then may become very complex and hard to code.
15	**Wiring layout clarity**	All wires secured, color-coded, labelled, and routed on designated channels, also protective sleaves.
16	**Ball containment integrity score**	100% containment — **no ball loss** during full run %90 range as 100% target may be unrealistic.
17	**Cube-fit compliance (400mm cubed)**	Entire robot must fit within a **400 mm cube** without compression or force.
18	**Ground clearance consistency**	Maintain **≥ 20 mm clearance** across all terrain transitions (e.g. see saw edges)
19	**End-of-run signal visibility**	LED or signal visible from **≥ 3 meters** at **120° viewing angle. (Green LED for indicating completion)**
20	**Safety features (no exposed wires moving parts)**	All moving parts and sharp edges fully enclosed or guarded to prevent user injury during operation
21	**Autonomous obstacle avoidance accuracy**	Robot detects and successfully avoids obstacles ≥ 90% of the time during the entire mission (With ultrasonic sensor)
22	**Communication interface reliability (Serial communication of the Arduino IDE)**	wired interface maintains uptime with no data loss during operation (Serial monitor in the code for troubleshooting of the code)
23	**Power supply protection**	Battery circuit includes inline fuse and surge protection to prevent damage during faults with a 20A fuse.
24	**Environmental robustness (dust/fire/water resistance)**	The use of protective sleaves on the wires protects the wires and also promotes neatness.
25	**Thermal management**	Operating temperature of electronics maintained below 60°C under continuous operation ***(There are heatsinks on the motor drivers to ensure that the circuits don't overheat)***
26	**Ease of coding and getting to the Arduino mega for code improvements.**	The USB cable will permanently be plugged into the Mega so that can be quickly plugged in for uploading code.

27	**Startup/reset readiness time**	Done by a single action such as flicking of a switch.
28	**Cost target**	Total build cost ≤ $400 AUD excluding labour

Appendix B

Detailed Calculations and Code Snippets

Contains any key mathematical calculations done in the code and annotated snippets of Arduino code for each major subsystem (drive, arm, blade, edge detection, etc.). Useful for validating the design decisions and functionality described in Section 5.5. All annotation in the code explains everything.

A-MainCode_R2spin2_version9_ULTRASONIC.ino

```
/===Libraries===
#include <AccelStepper.h> // Library for controlling stepper motor
#include <Servo.h> // Library for controlling servo motors

// === LED variable ===
const int statusLEDPin = A8; // Status LED pin (used to show completion or system states)

// === Stepper Motor Setup ===
const int stepPin = 37; // Step signal pin for stepper driver
const int directionPin = 35; // Direction signal pin
const int enablePin = 33; // Enable pin to turn stepper on/off
const int limitSwitchPin = 31; // Limit switch for homing stepper

AccelStepper stepper(AccelStepper::DRIVER, stepPin, directionPin); // Create stepper object

// === Adjustable Parameters to Turn 90 degrees===
const int pwmPercent   = 75; // Base PWM speed as a percentage
const int turnLeftTime = 500;  // ~45° turn
const int forwardTime  = 100;  // Forward movement time
const int delayBetween = 100;  // Delay between actions

int currentStep = 0; // Tracks the current sub-action in RETURN_HOME
bool actionStarted = false; // Flags whether a sub-action has been initiated

// === State Machine ===
enum State {
  HOMING,
  MOVE_TO_POSITION,
  RUN_MOTORS_FORWARD,
  SLOW_DOWN_FOR_COLLECTION,
  WAIT_FOR_COLLECTION,
  RETURN_HOME,
  FINAL_STOP
};
State currentState = HOMING; // Initial state when system powers on

```

```
// === DC Motor Setup ===
int speedPercent = 75; // Default motor speed (percentage)
int pwmValue = map(speedPercent, 0, 100, 0, 255); // Convert percentage to 8-bit PWM value

// === DC Motor fast and slow
int pwmValueFast = map(30, 0, 100, 0, 255);   // Fast speed (used for initial burst or movement)
int pwmValueSlow = map(30, 0, 100, 0, 255);   // used as a buffer Slow speed (used for controlled collection phase)
bool motorsSlowed = false; // Flag to track if motors have already been slowed
//20 is too low was found to be too low to move the robot reliably


// L298N Motor Driver Pins
// LEFT SIDE MOTORS (Driver 1)
#define ENA1 3 // Enable pin for Left Front motor
#define IN1A 36 // Direction pin 1 for Left Front motor
#define IN2A 38 // Direction pin 2 for Left Front motor

#define ENB1 7 // Enable pin for Left Rear motor
#define IN1B 40 // Direction pin 1 for Left Rear motor
#define IN2B 42 // Direction pin 2 for Left Rear motor

// RIGHT SIDE MOTORS (Driver 2)
#define ENA2 2 // Enable pin for Right Front motor
#define IN1C 47 // Direction pin 1 for Right Front motor
#define IN2C 49 // Direction pin 2 for Right Front motor

#define ENB2 6 // Enable pin for Right Rear motor
#define IN1D 53 // Direction pin 1 for Right Rear motor
#define IN2D 51 // Direction pin 2 for Right Rear motor

// === Blade Motor ===
#define BLADE_PWM_PIN 44 // PWM pin to control blade motor speed
#define BLADE_EN_PIN 46 // Enable pin to turn blade motor ON/OFF
int BladeSpeed = 90; // Blade speed in percent (adjustable)
int pwmValue_2 = map(BladeSpeed, 0, 100, 0, 255); // Convert BladeSpeed to PWM value as a percentage map (0-255)

// === Servos ===
Servo doorServo, rampServo; // Servo objects for door and ramp mechanisms
const int doorServoPin = 11, rampServoPin = 12; // Pins connected to door and ramp servos

// =Door Servo Angles=
int doorClosedAngle = 5, doorOpenAngle = 110; // Angle for door fully closed & Angle for door fully open
// =Ramp Servo Angles=

int rampTiltUpAngle = 70, rampTiltDownAngle = 120; // Angle to raise ramp (pre-collection), Angle to tilt ramp for ball release
int rampRestAngle = 90; // Neutral resting angle for ramp when idle

// === Timing Variables ===
unsigned long actionStartTime = 0; // Tracks the start time of current timed action
bool dcStartedBack = false; // Flag to check if DC motors started moving backward

// === Substate Tracking for RETURN_HOME Sequence ===
int returnHomeSubState = 0; // Tracks sub-phases of return-to-home logic

// === DC Motor Hold Power ===
int holdPowerPercent = 45; // Power percentage to hold DC motors in place (prevents rolling)
int pwmHoldValue = map(holdPowerPercent, 0, 100, 0, 255); // Convert hold power % to PWM value

```

```
// === Ultrasonic Sensor Setup (for edge detection) ===
const int trigPin = 26; // Trigger pin of HC-SR04 ultrasonic sensor
const int echoPin = 24; // Echo pin of HC-SR04 ultrasonic sensor
float distance = 0.0; // Measured distance in cm
unsigned long ultrasonicCheckInterval = 100;  // Interval to check distance (every 100 ms)
unsigned long lastUltrasonicCheck = 0; // Timestamp of last distance check
const int threshold = 60; // cm // Threshold distance in cm to detect edge/drop-off
bool edgeDetected = false; // Flag set true if edge is detected


//=====(START of void setup())=====
void setup() {
  Serial.begin(9600); // Initialize serial communication for debugging at 9600 baud

// === Stepper Motor Initialization ===
  stepper.setEnablePin(enablePin); // Assign enable pin for stepper driver
  stepper.setPinsInverted(false, false, true); // Configure step, direction, and enable pin logic inversion
  stepper.enableOutputs(); // Enable stepper motor outputs
  stepper.setAcceleration(1500); // Set acceleration (steps/s²)
  stepper.setMaxSpeed(1400); // Set maximum speed (steps/s)
  stepper.setSpeed(1400); // Set initial speed (steps/s)
  pinMode(limitSwitchPin, INPUT_PULLUP); // Configure limit switch pin as input with internal pull-up resistor in mega

// === DC Motor Driver Pins Setup ===
  pinMode(ENA1, OUTPUT); pinMode(IN1A, OUTPUT); pinMode(IN2A, OUTPUT); // Left Motor A pins
  pinMode(ENB1, OUTPUT); pinMode(IN1B, OUTPUT); pinMode(IN2B, OUTPUT); // Right Motor A pins
  pinMode(ENA2, OUTPUT); pinMode(IN1C, OUTPUT); pinMode(IN2C, OUTPUT); // Left Motor B pins
  pinMode(ENB2, OUTPUT); pinMode(IN1D, OUTPUT); pinMode(IN2D, OUTPUT); // Right Motor B pins

// === Blade Motor Pins Setup ===
  pinMode(BLADE_PWM_PIN, OUTPUT); // Blade PWM control pin
  pinMode(BLADE_EN_PIN, OUTPUT); // Blade enable pin
  digitalWrite(BLADE_EN_PIN, LOW); // Disable blade motor initially
  analogWrite(BLADE_PWM_PIN, 0); // Blade PWM duty cycle zero (off)

// === Ultrasonic Sensor Pins Setup ===
  pinMode(trigPin, OUTPUT);  // Ultrasonic sensor trigger pin as output
  pinMode(echoPin, INPUT); // Ultrasonic sensor echo pin as input

// === Status LED Setup ===
  pinMode(statusLEDPin, OUTPUT); // Status LED pin as output
  digitalWrite(statusLEDPin, LOW); // Turn status LED off initially

// === Servo Attachments and Initial Positions ===
  doorServo.attach(doorServoPin); // Attach door servo to pin
  rampServo.attach(rampServoPin);  // Attach ramp servo to pin
  doorServo.write(doorClosedAngle); // Set door servo to closed position
  rampServo.write(rampRestAngle); // Set ramp servo to resting (neutral) position
  delay(500);  // Wait 500 ms to allow servos to reach position
}
//=====(END of void setup())=====

```

```
// === (void) Ramp Control Functions ===
// Tilt ramp up to allow balls to roll down for collection
void tiltRampUp() {
  rampServo.write(rampTiltUpAngle);
}

// Tilt ramp down to allow balls to be released
void tiltRampDown() {
  rampServo.write(rampTiltDownAngle);
}

// === (Void) Door Control Functions ===
// Open the door to release or allow balls to enter storage
void openDoor() {
  doorServo.write(doorOpenAngle);
  }

// Close the door to secure the balls inside the storage
void closeDoor() {
  doorServo.write(doorClosedAngle);
  }

// === (Void) Movement Function ===

// Controls all 4 drive motors (2 left, 2 right) to move forward or backward
// 'forward = true' will move robot forward, 'forward = false' will move it backward
void moveAllMotors(bool forward) {
  int a = forward ? HIGH : LOW; // Set direction pin A
  int b = forward ? LOW : HIGH; // Set direction pin B (opposite of A)
  // Apply direction to all 4 motor channels
  digitalWrite(IN1A, a); digitalWrite(IN2A, b); // Left Motor A
  digitalWrite(IN1B, a); digitalWrite(IN2B, b); // Right Motor A
  digitalWrite(IN1C, a); digitalWrite(IN2C, b);  // Left Motor B
  digitalWrite(IN1D, a); digitalWrite(IN2D, b); // Right Motor B
  // Set speed using PWM to all EN pins (enable)
  analogWrite(ENA1, pwmValue); analogWrite(ENB1, pwmValue);
  analogWrite(ENA2, pwmValue); analogWrite(ENB2, pwmValue);
  /*
  analogWrite(ENA1, pwmValue);  // Left Front Motor (Driver 1)
  analogWrite(ENB1, pwmValue);  // Right Front Motor (Driver 1)
  analogWrite(ENA2, pwmValue);  // Left Rear Motor (Driver 2)
  analogWrite(ENB2, pwmValue);  // Right Rear Motor (Driver 2)
  */
}
```

```
191
192   // === (Void) Stop all DC motors by setting PWM to 0 (no power) ===
193   void stopAllMotors() {
194     analogWrite(ENA1, 0); analogWrite(ENB1, 0);
195     analogWrite(ENA2, 0); analogWrite(ENB2, 0);
196     /*
197     analogWrite(ENA1, 0); // Stop Left Front Motor
198     analogWrite(ENB1, 0); // Stop Right Front Motor
199     analogWrite(ENA2, 0); // Stop Left Rear Motor
200     analogWrite(ENB2, 0); // Stop Right Rear Motor
201     */
202   }
203
```

```
204   // === (Void)Turn Robot Right ===
205   // Left motors move forward, right motors move backward to pivot the robot clockwise
206   void turnMotorsRight() {
207     // Left Front & Rear Motors (A & C) forward
208     digitalWrite(IN1A, HIGH); digitalWrite(IN2A, LOW);
209     digitalWrite(IN1B, HIGH); digitalWrite(IN2B, LOW);
210
211     // Right Front & Rear Motors (B & D) backward
212     digitalWrite(IN1C, LOW); digitalWrite(IN2C, HIGH);
213     digitalWrite(IN1D, LOW); digitalWrite(IN2D, HIGH);
214
215     // Apply PWM power to all motors
216     analogWrite(ENA1, pwmValue); analogWrite(ENB1, pwmValue);
217     analogWrite(ENA2, pwmValue); analogWrite(ENB2, pwmValue);
218   }
219
220   // === (Void function) Turn Robot Left ===
221   // Right motors move forward, left motors move backward to pivot the robot counter-clockwise
222   void turnMotorsLeft() {
223     // Left Front & Rear Motors (A & C) backward
224     digitalWrite(IN1A, LOW); digitalWrite(IN2A, HIGH);
225     digitalWrite(IN1B, LOW); digitalWrite(IN2B, HIGH);
226
227   // Right Front & Rear Motors (B & D) forward
228     digitalWrite(IN1C, HIGH); digitalWrite(IN2C, LOW);
229     digitalWrite(IN1D, HIGH); digitalWrite(IN2D, LOW);
230
231   // Apply PWM power to all motors
232     analogWrite(ENA1, pwmValue); analogWrite(ENB1, pwmValue);
233     analogWrite(ENA2, pwmValue); analogWrite(ENB2, pwmValue);
234   }
235
236   // === (Void function) Hold Robot in Place ===
237   // All motors receive low forward torque to resist movement and not slip on see saw
238   void holdAllMotors() {
239     // All motors set to forward direction
240     digitalWrite(IN1A, HIGH); digitalWrite(IN2A, LOW);
241     digitalWrite(IN1B, HIGH); digitalWrite(IN2B, LOW);
242     digitalWrite(IN1C, HIGH); digitalWrite(IN2C, LOW);
243     digitalWrite(IN1D, HIGH); digitalWrite(IN2D, LOW);
244     // Apply reduced PWM to maintain position (prevent rollback on slopes)
245     analogWrite(ENA1, pwmHoldValue); analogWrite(ENB1, pwmHoldValue);
246     analogWrite(ENA2, pwmHoldValue); analogWrite(ENB2, pwmHoldValue);
247   }
248
```

```
249   // === (Void function) Read Distance from Ultrasonic Sensor ===
250   // Sends a trigger pulse and calculates the distance to the nearest object based on echo return time.
251   float readUltrasonicDistance() {
252     // Ensure trigger pin is LOW for at least 2 µs before sending the HIGH pulse
253     digitalWrite(trigPin, LOW);
254     delayMicroseconds(2);
255      // Send a 10 µs HIGH pulse to trigger the ultrasonic burst
256     digitalWrite(trigPin, HIGH);
257     delayMicroseconds(10);
258     digitalWrite(trigPin, LOW);
259     // Measure the duration of the incoming pulse on the echo pin (max 20 ms timeout)
260     float duration = pulseIn(echoPin, HIGH, 20000UL); // Timeout after 20,000 µs (20 ms)
261     // Convert time (µs) to distance (cm): speed of sound = 343 m/s = 0.0343 cm/µs
262     // Divide by 2 to account for round-trip time (out and back)
263     return (duration * 0.0343) / 2;
264   }
265
```

```
void loop() {
  switch (currentState) {

    // === HOMING STATE ===
    // This state moves the stepper motor backwards until it hits the limit switch,
    // establishing a known "home" position for the stepper motor.
    case HOMING:
      tiltRampUp(); // Raise the ramp to its default (collection ready) position
      stepper.moveTo(stepper.currentPosition() - 700); // Command the stepper motor to move slightly backwards
      stepper.run(); // Non-blocking call to move motor toward the target

      // Check if the limit switch has been triggered (LOW = activated)
      if (digitalRead(limitSwitchPin) == LOW) {
        stepper.setCurrentPosition(0);  // Reset current position to 0 (home)
        stepper.moveTo(40100); // Set target position for collection zone
        currentState = MOVE_TO_POSITION; // Transition to next state
      }
      break;

    // === MOVE_TO_POSITION STATE ===
    // Moves the stepper motor forward to the pre-defined collection location.
    // Once the stepper reaches the target, the collection phase begins.
    case MOVE_TO_POSITION:
      stepper.run(); // Continue moving stepper to target position (non-blocking)

      // Once stepper reaches target position (i.e., distanceToGo == 0)
      if (stepper.distanceToGo() == 0) {
        openDoor(); // Open the collection door to allow balls to enter
        digitalWrite(BLADE_EN_PIN, HIGH);  // Start the blade motor for assisting ball collection
        analogWrite(BLADE_PWM_PIN, pwmValue_2); // Set blade speed
        pwmValue = pwmValueFast; // Set DC motors to fast mode initially
        moveAllMotors(true); // Begin forward movement of DC motors
        actionStartTime = millis(); // Store the time to track delay before slowing down
        motorsSlowed = false;  // Reset slowdown flag for next state
        currentState = SLOW_DOWN_FOR_COLLECTION; // Proceed to slowdown state
      }
      break;
```

```
    // === SLOW_DOWN_FOR_COLLECTION STATE ===
    // After an initial burst of speed, this state reduces the drive motor speed
    // to ensure controlled ball collection and improved accuracy.
    case SLOW_DOWN_FOR_COLLECTION:
      // Wait at least 250ms after initial motor start before slowing down
      if (!motorsSlowed && millis() - actionStartTime >= 250) {
        pwmValue = pwmValueSlow;        // Reduce motor speed to slow collection speed
        moveAllMotors(true);            // Re-apply movement at reduced speed
        motorsSlowed = true;            // Flag to avoid slowing multiple times
        actionStartTime = millis();     // Reset timer for use in next state
        currentState = RUN_MOTORS_FORWARD; // Move to next state: active collection and edge detection
      }
      break;

    // === RUN_MOTORS_FORWARD STATE ===
    // Robot drives forward at slower speed while checking for edge using ultrasonic sensor.
    // If edge is detected or 2 seconds have passed, it stops to begin collection.
    case RUN_MOTORS_FORWARD:
      // Check ultrasonic sensor at regular intervals
      if (millis() - lastUltrasonicCheck >= ultrasonicCheckInterval) {
        lastUltrasonicCheck = millis(); // Update last check time
        distance = readUltrasonicDistance(); // Measure distance to ground
        Serial.print("Distance: ");
        Serial.println(distance);
        // If valid distance and edge detected (i.e., sudden drop), flag it
        if (distance > threshold && distance < 300 && distance > 0) {
          edgeDetected = true;
        }
      }
      // Proceed to next state if edge is detected or timeout reached
      if (edgeDetected || millis() - actionStartTime >= 2000) {
        stopAllMotors(); // Stop all DC motors
        digitalWrite(BLADE_EN_PIN, HIGH); // Enable blade motor
        analogWrite(BLADE_PWM_PIN, pwmValue_2); // Set blade motor speed
        currentState = WAIT_FOR_COLLECTION; // Transition to waiting state
        actionStartTime = millis(); // Reset timer
        edgeDetected = false; // Reset edge detection flag
      }
      break;
```

```
344     // === WAIT_FOR_COLLECTION STATE ===
345     // Robot waits 8 seconds for the blade to collect balls, then prepares to return home.
346     case WAIT_FOR_COLLECTION:
347       // Wait for the full 8-second collection duration
348       if (millis() - actionStartTime >= 8000) {
349         closeDoor();  // Close storage door to secure collected balls
350         digitalWrite(BLADE_EN_PIN, LOW); // Turn off blade motor
351         analogWrite(BLADE_PWM_PIN, 0); // Ensure blade motor PWM is zero
352         stepper.moveTo(0); // Command stepper to return to home (starting position)
353         actionStartTime = millis(); // Reset timer
354         dcStartedBack = false; // Reset return movement flag
355         speedPercent = 75; // Reset speed to default
356         pwmValue = map(speedPercent, 0, 100, 0, 255); // Recalculate PWM value
357         currentState = RETURN_HOME; // Transition to RETURN_HOME state
358       }
359       break;
360
```

```
361     // === RETURN_HOME STATE ===
362     // Controls the return path of the robot using a substate machine for complex maneuvers
363     case RETURN_HOME:
364       stepper.run(); // Keep stepper motor running toward home position
365       switch (returnHomeSubState) {
366
367         case 0:  // Initial delay before moving DC motors backward
368           if (millis() - actionStartTime >= 9000) { // Wait 9 seconds after previous actions
369             returnHomeSubState = 1;  // Ensures that the colleciton vain is above see saw clearance
370           }
371           break;
372
373         case 1: // Move all DC motors backward (or reverse direction)
374           moveAllMotors(false);  // Move backward or appropriate direction
375           actionStartTime = millis(); // Reset timer for next step
376           returnHomeSubState = 2;
377           break;
378
379         case 2: // Stop motors after 500 ms reverse movement
380           if (millis() - actionStartTime >= 500) {
381             stopAllMotors();
382             actionStartTime = millis();
383             returnHomeSubState = 3;  // Proceed to complex path maneuvers
384           }
385           break;
386
387         case 3: { // Complex maneuver sequence: multiple turns and forward moves
388           unsigned long now = millis();
389           switch (currentStep) {
390             case 0: // First turn (turn left)
391               if (!actionStarted) {
392                 Serial.println("Turn 1...");
393                 turnMotorsLeft();
394                 actionStartTime = now;
395                 actionStarted = true;
396               } else if (now - actionStartTime >= turnLeftTime) {
397                 stopAllMotors();
398                 actionStarted = false;
399                 currentStep++;
400               }
401               break;
402
```

```
403             case 1: // Move forward briefly after first turn
404               if (!actionStarted) {
405                 Serial.println("Forward 1...");
406                 moveAllMotors(true);
407                 actionStartTime = now;
408                 actionStarted = true;
409               } else if (now - actionStartTime >= forwardTime) {
410                 stopAllMotors();
411                 actionStarted = false;
412                 currentStep++;
413               }
414               break;
415
416             case 2: // Second turn (turn left again) achieves a 90 degree turn
417               if (!actionStarted) {
418                 Serial.println("Turn 2...");
419                 turnMotorsLeft();
420                 actionStartTime = now;
421                 actionStarted = true;
422               } else if (now - actionStartTime >= turnLeftTime) {
423                 stopAllMotors();
424                 actionStarted = false;
425                 currentStep++;
426               }
427               break;
428
```

```
429             case 3: // Move forward again to complete return path maneuver
430               if (!actionStarted) {
431                 Serial.println("Forward 2...");
432                 moveAllMotors(true);
433                 actionStartTime = now;
434                 actionStarted = true;
435               } else if (now - actionStartTime >= forwardTime) {
436                 stopAllMotors();
437                 Serial.println("Return path complete.");
438                 actionStarted = false;
439                 currentStep++;
440               }
441               break;
442
443             case 4:
444             // Finished maneuver steps, advance substate
445               returnHomeSubState = 4;  // Proceed to next substate
446               break;
447           }
448
449           break;
450         }
451
452           case 4: // Pause for 1 second after maneuvers
453             stopAllMotors();
454             if (millis() - actionStartTime >= 1000) {
455               actionStartTime = millis();
456               returnHomeSubState = 5;
457             }
458             break;
459
```

```
460         case 5: // Move forward for 2.8 seconds
461           moveAllMotors(true);
462           if (millis() - actionStartTime >= 2800) {
463             stopAllMotors();
464             actionStartTime = millis();
465             returnHomeSubState = 6;
466           }
467           break;
468
469         case 6: // Pause for 3 seconds
470           stopAllMotors();
471           if (millis() - actionStartTime >= 3000) {
472             actionStartTime = millis();
473             returnHomeSubState = 7;
474           }
475           break;
476
477         case 7: // Short forward move for 150 ms
478           moveAllMotors(true);
479           if (millis() - actionStartTime >= 150) {
480             stopAllMotors();
481             actionStartTime = millis();
482             returnHomeSubState = 8;
483           }
484           break;
485
486         case 8: // Turn left for 1 second
487           turnMotorsLeft();
488           if (millis() - actionStartTime >= 1000) {
489             stopAllMotors();
490             actionStartTime = millis();
491             returnHomeSubState = 9;
492           }
493           break;
494
495         case 9: // Move forward for 500 ms
496           moveAllMotors(true);
497           if (millis() - actionStartTime >= 500) {
498             stopAllMotors();
499             actionStartTime = millis();
500             returnHomeSubState = 10;
501           }
502           break;
503
```

```
      case 10: // Prepare ramp and door for ball release
        stopAllMotors();
        tiltRampDown(); // Lower ramp
        openDoor(); // Open door to release balls
        if (millis() - actionStartTime >= 1000) {
          closeDoor();  // Close door after release
          moveAllMotors(false);  // Move backward slightly
          actionStartTime = millis();
          returnHomeSubState = 11;
        }
        break;

      case 11:  // Wait 300 ms
        if (millis() - actionStartTime >= 300) {
          stopAllMotors();
          actionStartTime = millis();
          returnHomeSubState = 12;
        }
        break;

      case 12:  // Wait 300 ms, ensure door closed, check limit switch for homing
        if (millis() - actionStartTime >= 300) {
          closeDoor();
          actionStartTime = millis();
          returnHomeSubState = 13;
        }
        if (digitalRead(limitSwitchPin) == LOW) {  // Limit switch triggered (home reached)
          stepper.stop();
          stepper.setCurrentPosition(0); // Reset stepper position
          returnHomeSubState = 0; // Reset substate for next cycle
          currentStep = 0; // Reset currentStep as well
          currentState = FINAL_STOP; // Transition to final stop state
        }
        break;

      case 13: // Final stepper stop and indicate completion
        stepper.stop();
        digitalWrite(statusLEDPin, HIGH); // Turn on status LED to signal completion
        break;

      default:
        // Optionally reset
        break;
    }

    // Safety check for limit switch triggered outside substate 12
    if (digitalRead(limitSwitchPin) == LOW) {
      stepper.stop();
      stepper.setCurrentPosition(0);
      returnHomeSubState = 0;
      currentStep = 0; // Reset currentStep as well
      currentState = FINAL_STOP;
      digitalWrite(statusLEDPin, HIGH); // Indicate completion // Turn on status LED
    }
    break;

  // === FINAL_STOP STATE ===
  // Stops all motors and indicates completion with LED
  case FINAL_STOP:
    stopAllMotors();
    digitalWrite(statusLEDPin, HIGH); // Indicate completion Turn on status LED
    break;

  default:
    // Optional: handle unknown or error states
    break;
 }
}
```

***The code has Annotations that explain what each part does exactly.**

Appendix C

Test Plans, Raw Data, and Analysis

Includes raw data tables, trial logs, graphs (e.g., task completion times), and testing procedures as described in the Device Evaluation chapter.

Supports the summary table and graphs referenced in Section 6 of the report.

C.1: Multimeter test

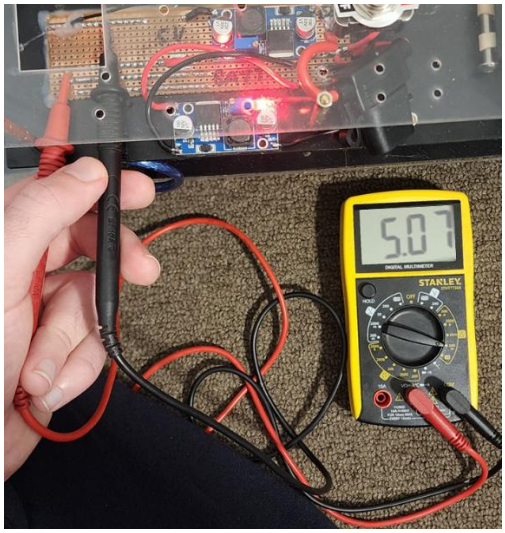

***Figure C.1**: Testing the amount of Amps drawn when all components switched on and working, around 4-5 Amps*

**note: the average amps that was measured is approximately 2A when running the system with the obstacle course code. Also that Multimeter is set to measure amps.*

C.2: Task completion timed trails (R2-spin2 reaches the endpoint)

***Table C.1:** Task Completion Times Recorded Over 15 Trials Using Video Analysis*

Trial Number	Task Completion Time (seconds)
1	89
2	77
3	90
4	98
5	76
6	75
7	74

Trial Number	Task Completion Time (seconds)
9	70
10	70
11	76
12	65
13	70
14	76
15	78
Average	77

A meticulous time analysis was carried out over 15 trials in order to assess the robot's overall performance in finishing the competition course. Every trial was captured on video, and the job completion time was calculated by closely examining each clip to determine the precise times the robot started moving (with the "Start Clap") and stopped completely at the conclusion of its controlled operation. The average completion time was **77 seconds**, with documented completion times ranging **from 65 to 98 seconds** *(see Table C.1)*. The system's potential under ideal circumstances was demonstrated by Trial 12's highest performance, which was recorded at **65 seconds**, but had issues with the ramp as the robot would move across the see saw a bit too fast risking injury to the robot R2Spin2. Variations in initial ball placement, wheel traction, and ambient factors like illumination that alter sensor readings were all blamed for minor time variances among trials. The trend of improvement in later trials suggests that as the system was further refined and tuned, its consistency and speed of task execution improved significantly. Overall, the robot comfortably meets the RUNTIME specification threshold of completing the task within 120 seconds, demonstrating a strong and reliable performance under repeated testing. Without regard to all the balls being deposited.

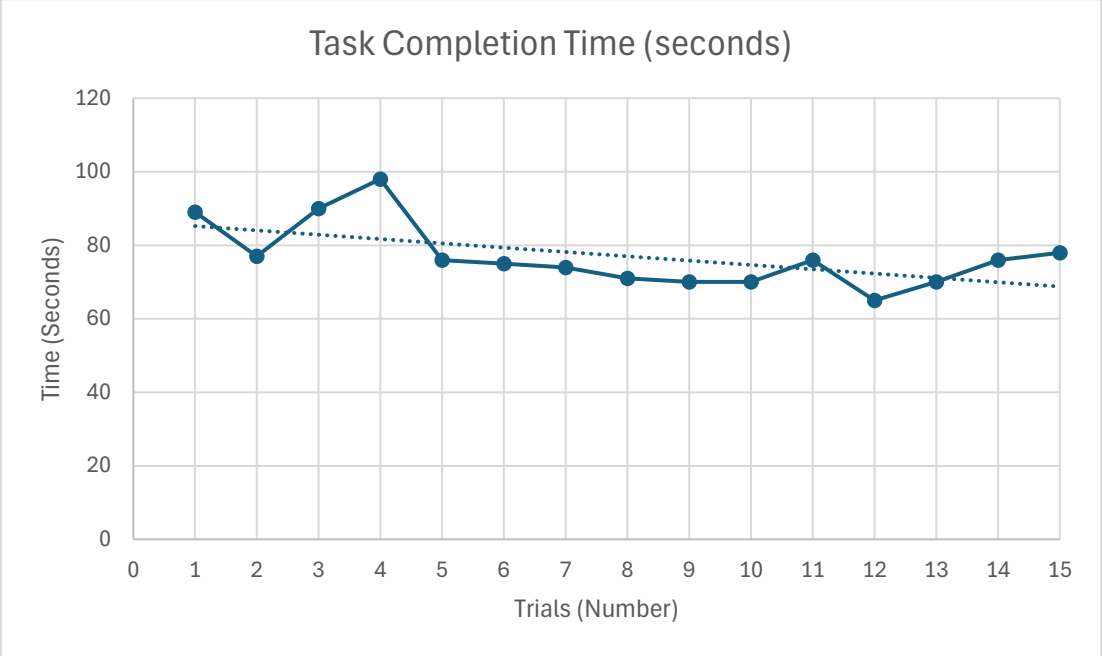

Figure C.2: *Line Graph of Task Completion Time Across 15 Trials*

C.3 Ball Collection accuracy

With the same 15 trials the balls collected where also recorded as per the table below:

Tape was added on the servo door to ensure the balls stayed in, this improved results

Table C.3: *Recorded trails and ball collection accuracy*

Trial #	Tennis Ball	Racquetball	Table Tennis Ball	Total Collected	Success Rate (%)
1	✓	✓	✓	3	100%
2	✓	✓	x	2	66.70%
3	✓	x	✓	2	66.70%
4	✓	✓	✓	3	100%
5	✓	✓	x	2	66.70%
6	x	✓	✓	2	66.70%
7	✓	✓	✓	3	100%
8	✓	x	✓	2	66.70%
9	✓	✓	x	2	66.70%
10	✓	✓	✓	3	100%
11	✓	x	✓	2	66.70%
12	x	✓	✓	2	66.70%
13	✓	x	✓	2	66.70%
14	✓	✓	x	2	66.70%
15	✓	x	✓	2	66.70%
Average					**76%**
Total	**13**	**10**	**11**	**34**	

Appendix D

Manufacturing, CAD, and Assembly Documentation

Includes final CAD, views of subsystems, 3D print STL references, and images of prototypes at various stages. This also includes photos of the progress of the build process which was very complex and meticulous and took lots of time.

Referenced throughout Section 5.1–5.4 for physical design validation.

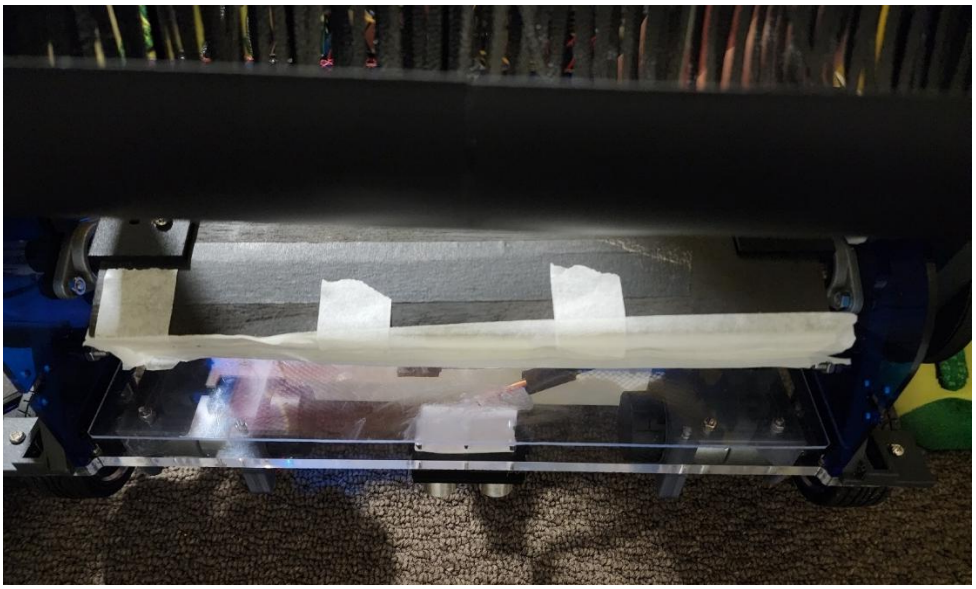

Figure D.1: *Masking tape on front door*

Figure D.2: *Tapping drill bit*

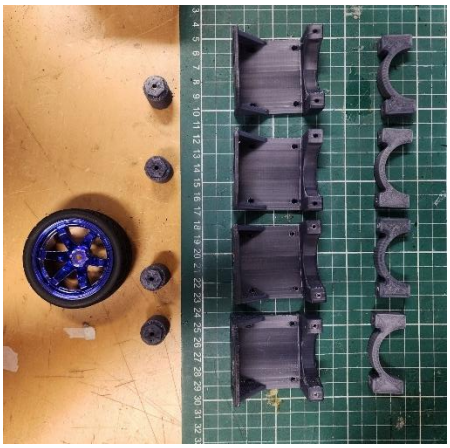

Figure D.3: *Figure 7 – Final design 3D printed motor mounts with wheel attachments.*

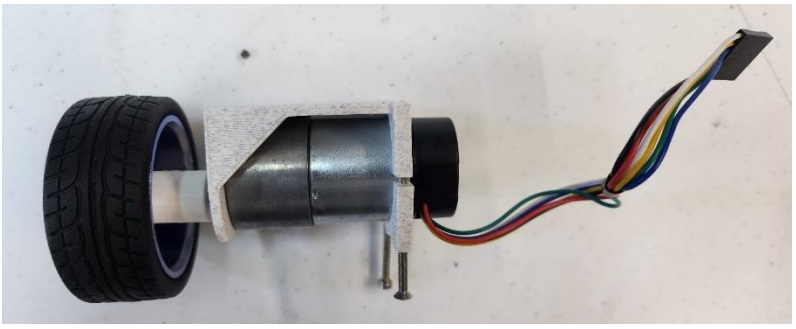

Figure D.4: *Test print 3D printed motor mount with wheel attachments.*

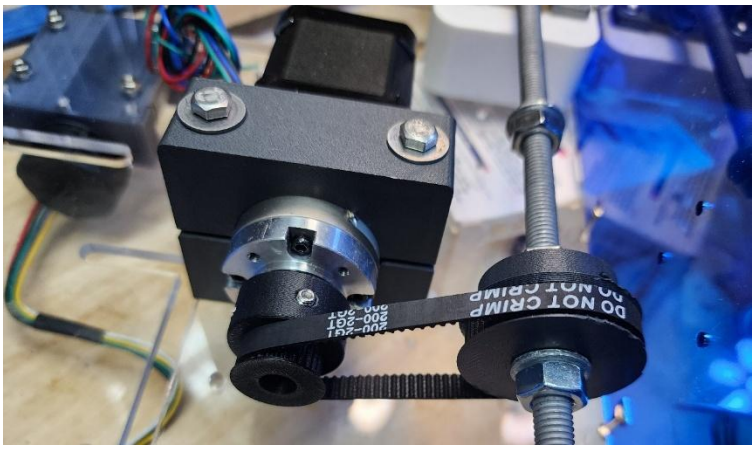

Figure D.5: *GT2 timing belt pully on the Nema 17 stepper motor and middle rod*

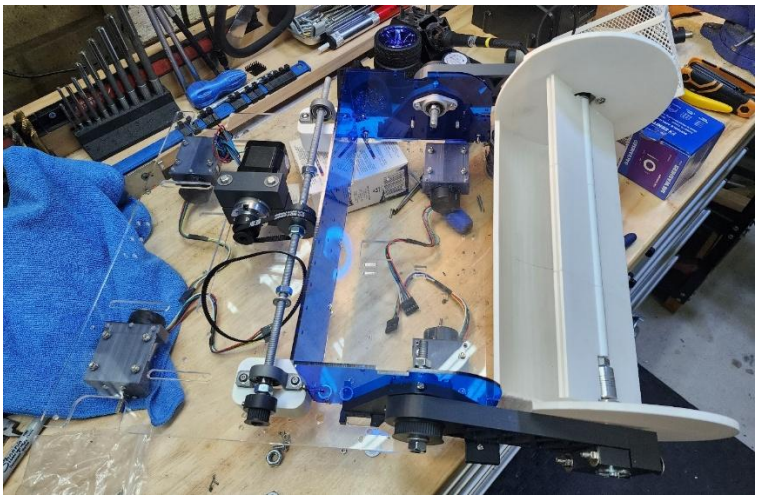

Figure D.6: *Progress build of the final design.*

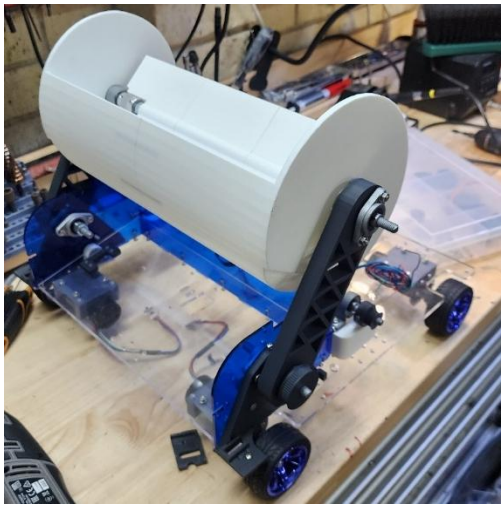

Figure D.7: *Progress build of the final design*

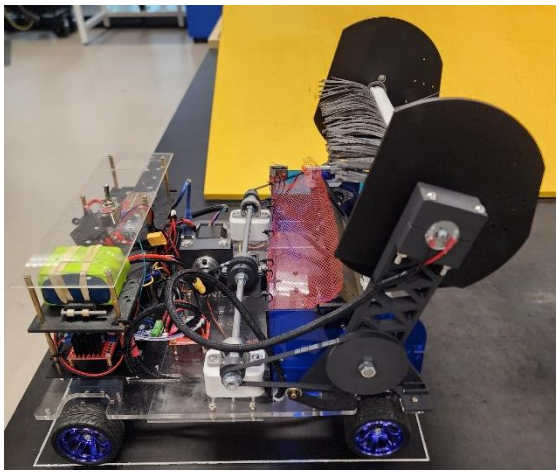

Figure D.8*: Final design side view on the course.*

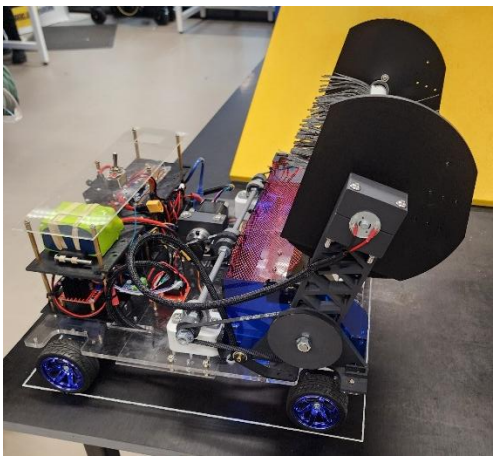

Figure D.9*: Final design side view on the course other view.*

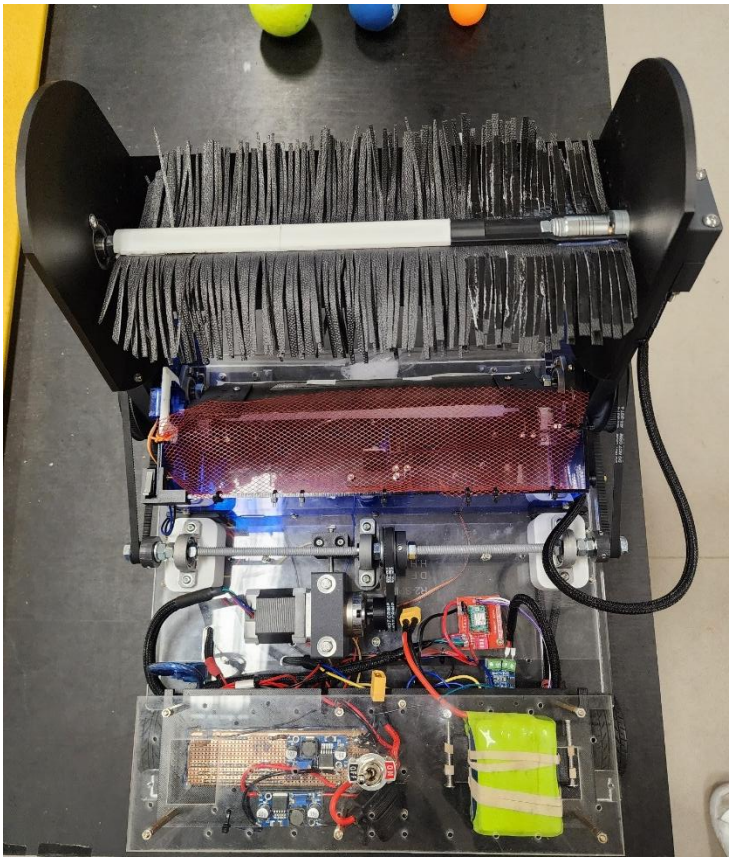

Figure D.10*: Final design top view on the course.*

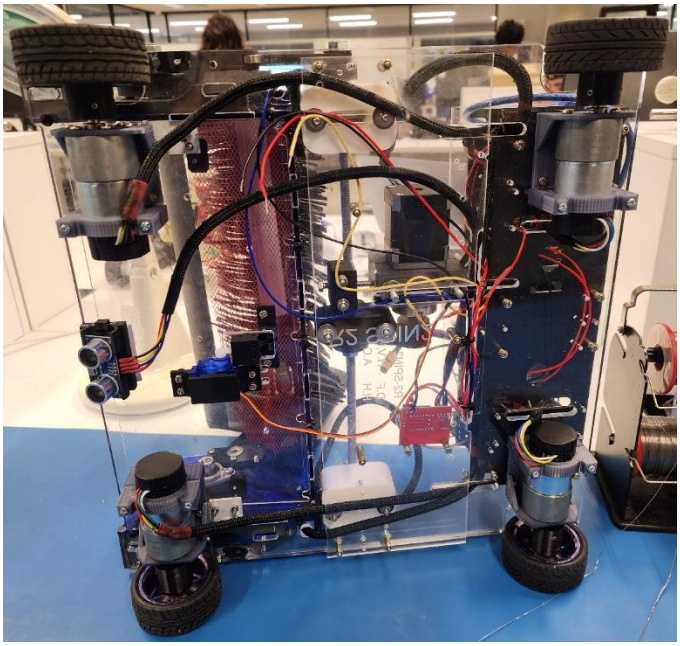

Figure D.11*: Final design bottom view.*

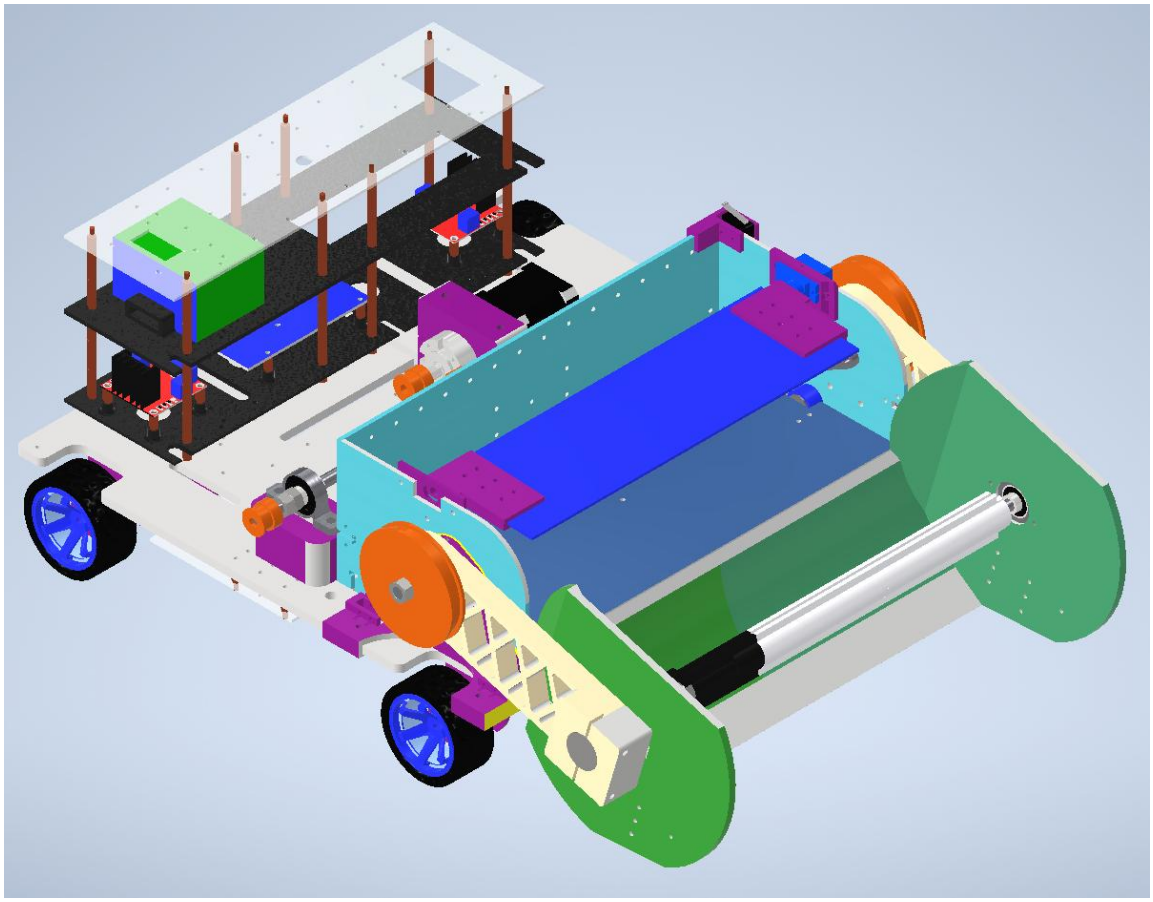

Figure D.12*: Final design CAD Assembly model home view.*

Parts like **the tensioner (5.3 Collection System) was added later after realising that the tensioner was needed to ensure that the GT2 timing belt was tensioned enough and not sloppy. Also the **ultrasonic sensor mount** at the front of the bot was a last minute alteration.*

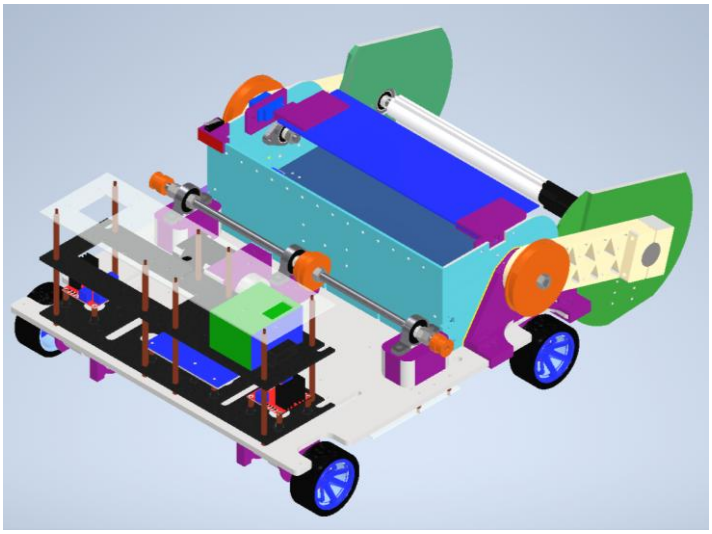

Figure D.13*: Final design CAD Assembly model back view right.*

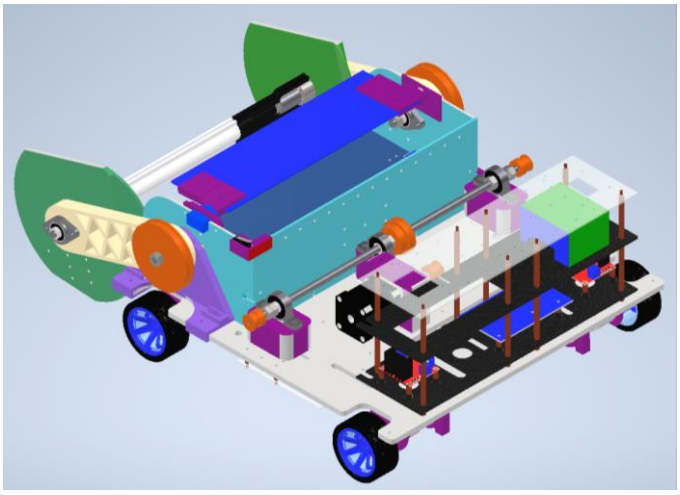

Figure D.14*: Final design CAD Assembly model back view left view.*

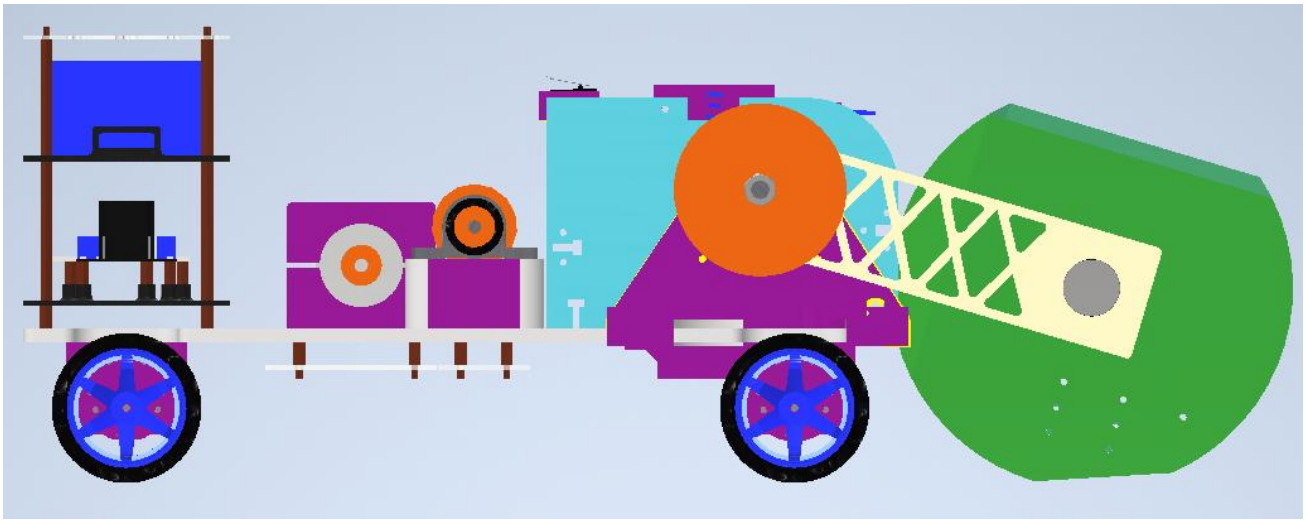

Figure D.15*: Final design CAD Assembly model side view right.*

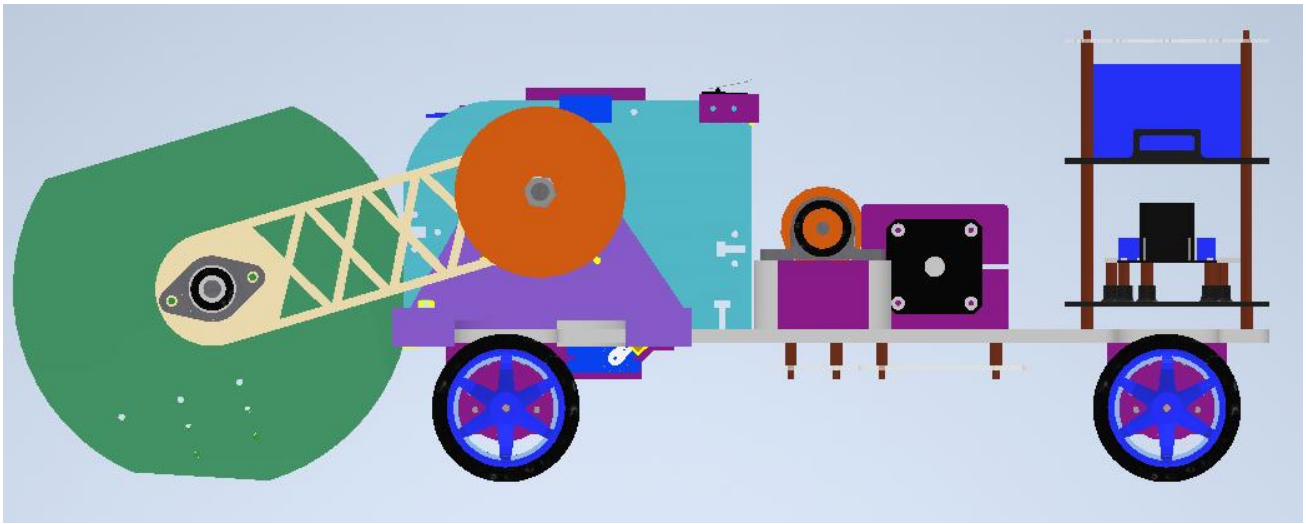

Figure D.16*: Final design CAD Assembly model side view left*

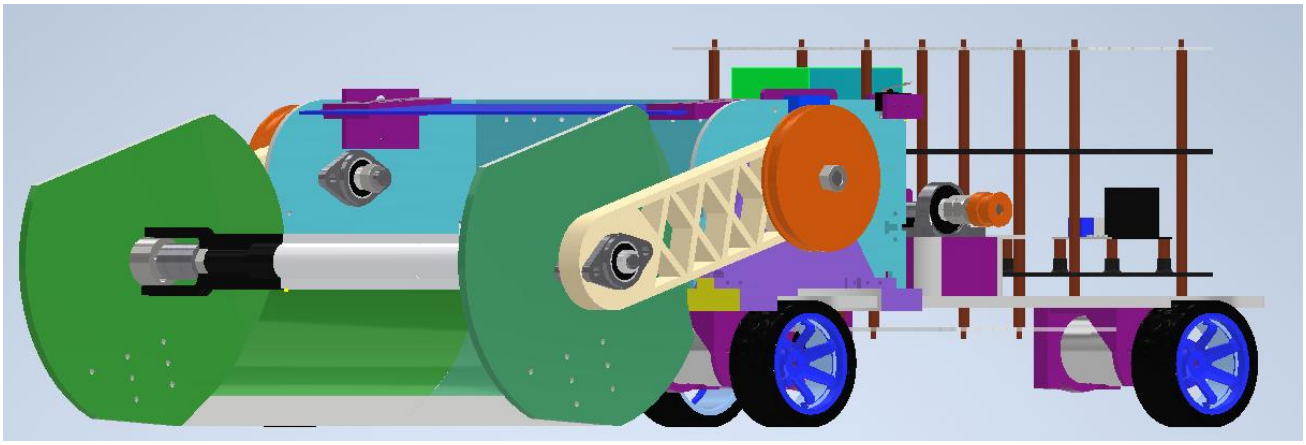

Figure D.17*: Final design CAD Assembly model side from front.*

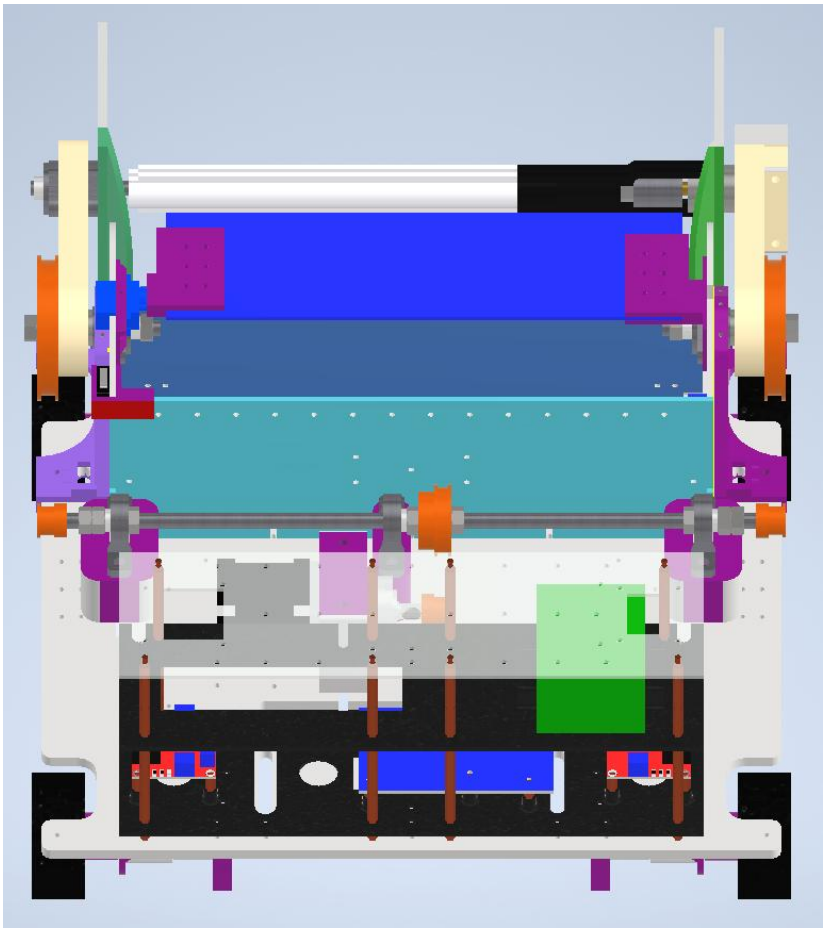

Figure D.18*: Final design CAD Assembly model back top view.*

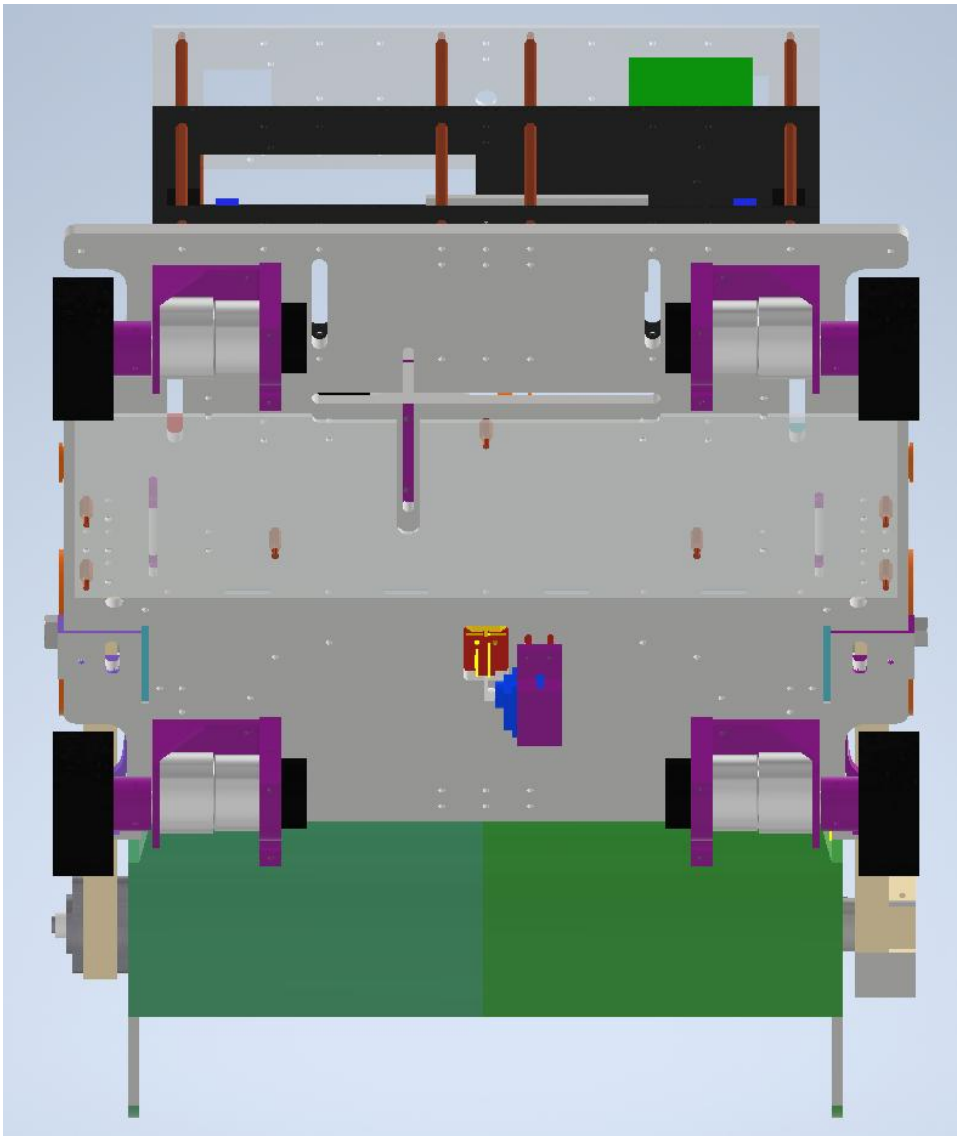

Figure D.19*: Final design CAD Assembly bottom view slanted.*

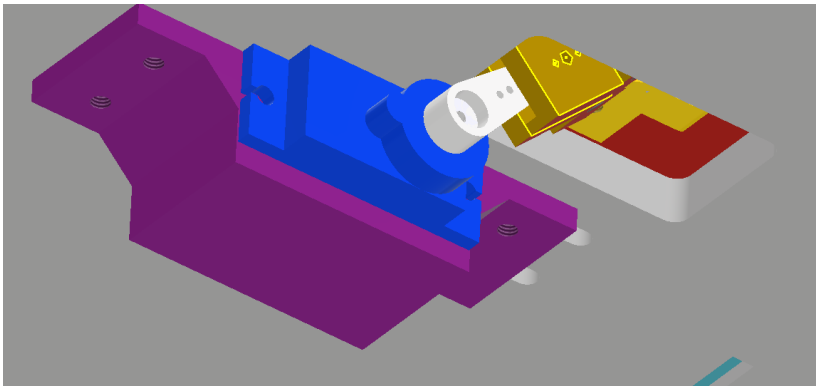

Figure D.20*: Final design CAD Assembly servo motor mount on the bottom for ramp.*