BuzzRover

Autonomous Transport for Sustainable Beekeeping

Team Members

Bora Chen
Joginder Dhaliwal
Duke Liu
Jad Mhanna
Lusako Mwakatobe
Jamie Peacock
Neel Shah
Ruvan Sidrake
Khelan Tailor
Sutong Zhang

Abbey Park High School

In Collaboration With:
Willow Grove Heritage
Bee & Egg Farm

AgRobotics Student Challenge
March 5th, 2026

v1.0 - Technical Document

TABLE OF CONTENTS

OVERVIEW

Our community partner is Willow Grove Heritage Bee & Egg Farm, a woman-led agricultural business operated by Karin Tomosky in Milton, Ontario. The farm manages 20 honeybee hives producing 350L of honey per year alongside an egg operation, with plans to expand in the future. Each honey super holds nine honey frames and weighs 60 - 80 lbs. Transporting the honey supers from the apiary to the indoor extraction room is a manual process using a cart. Our goal was to design a rover that improves efficiency and reduces physical workload: the BuzzRover.

Bee farming is crucial in Canada for two main reasons. Firstly, it provides essential pollination services for a large range of crops. These pollination services are very important for the agricultural industry. The second reason is that the honey produced by bees and other bee products generate revenue and contribute to domestic and international markets. For example, “Managed honey bees pollinate 80% of all agricultural crops requiring insect pollination. They account for $395 million in pollination services to Ontario farmers and contribute $30 million a year in honey sales” (Government of Canada). The beekeeping industry ultimately helps and drives the success of other agricultural industries, providing pollination services to farmers of orchard fruits, berries, vegetables, forage, and canola seeds.

The BuzzRover is a remote controlled transport rover designed to move honey frames to the extraction room in a safe, efficient manner. The rover uses off-road rubber wheels with tread patterns for strong traction on grass, dirt, and uneven terrain, ensuring stability while carrying heavy loads. The rover’s chassis is built from lightweight aluminum, which is strong, corrosion resistant, unlikely to warp. Aluminum is also easy to machine, drill, or rivet, making it ideal for precise, durable and modular construction. Using brushless DC motors, the drivetrain is able to deliver smooth torque, high efficiency, and a long operational life alongside minimal noise. The rover will be powered by nickel metal hydride batteries. NiMH batteries have the greatest heat tolerance as they are operational up to 65 degrees Celsius, making them capable of withstanding the summer heat. Compared with some lithium-based packs, NiMH cells are generally more tolerant of occasional abuse such as brief high-current draws, vibration, and minor over-discharge events,

which is useful in an agricultural environment (Eletek, 2025). The rover has a low center of gravity which improves safety by preventing tipping during transport. The rover will also be modular, allowing additions such as tool racks, an expanded frame carrier, or a honey frame lifter. Similarly, maintenance is simplified through standard fasteners, swappable batteries, independent motor pods, and an access panel for servicing. Lastly, we've included weatherproof connectors, rubber dampers, and sealed housing for a long lifespan in outdoor conditions. This solution reduces physical strain, protects honey frames during transport, and improves overall workflow efficiency on the farm.

COMMUNITY PARTNER

The Willow Grove Heritage Honey Bee and Egg Farm is a local leader in sustainable heritage farming. Originally built on a historical heritage site, it has been transformed into a haven for ducks and bees. Their dedication to the land inspires our dedication to engineering and together we're exploring how the next generation of technology can advance farm logistical demands.

Our agriculture-based community partner, Karin, is a dedicated beekeeper who developed her expertise through formal training at the University of Guelph, where she studied bee biology and hive management. She passionately cares for her bees year-round with each season having a set of tasks she must complete. In the fall, she prepares the bees for the winter. She reduces hive entrances to prevent pests and rodents from invading, ensures the bees have sufficient stored honey to sustain them through the colder months, and adds insulation when necessary. Since winter survival directly impacts spring production, these preparations become necessary.

COMMUNITY PARTNER PROBLEM

During the summer the bees are at their peak activity of collecting nectar and producing honey, which is when the harvesting starts. During harvesting, Karin removes heavy frames filled with honey and uses an extractor to spin out the honey before bottling it into 250 mL and 500 mL jars. However, transporting these frames from the hives to the extraction room is physically demanding and time-consuming. To support her honey production process, we designed a rover that helps transport honey frames to the extraction room. This reduces physical strain, improves efficiency during harvest, and allows Karin to focus more on hive health and production.

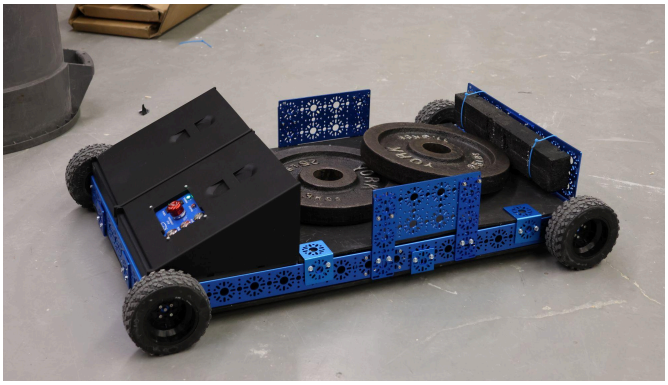
THE PROBLEM AND DESIGN REQUIREMENTS

Bee Farms at a small scale often rely on the manual transport of honey supers, which can weigh in excess of 60lbs when full of honey. This creates the risk of injury or damage to the product while being physically demanding and time-consuming. This is where we can help, with a rover designed for the challenges of an agricultural environment. It's designed for uneven terrain, outdoor exposure, and the need to operate quietly and safely around bees. These conditions guided all design decisions for BuzzRover.

RESEARCH

MOBILITY

The BuzzRover uses four rubber wheel tires powered by 4 motors to provide stable and efficient mobility. Rubber is a flexible and highly durable material that can endure different terrains and withstand driving over long distances for years (“Rubber Tires: Mastering the Artistry of Every Dynamic Ride”, 2024). The flexibility and longevity of rubber help the robot maintain control while transporting loads across lengthy, uneven terrain through the farm.

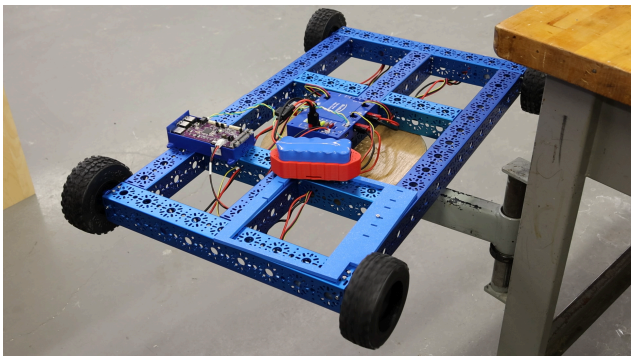


With four points of ground contact, the system distributes weight effectively and enhances overall stability, particularly when operating over slopes or soft ground. Additionally, using four independently driven motors enables skid-steer turning, allowing the BuzzRover to turn in place without a complex steering mechanism. This wheeled system is lightweight, energy-efficient, and easier to maintain, making it well-suited for agricultural field conditions.

DRIVE MOTORS AND CONTROLLER

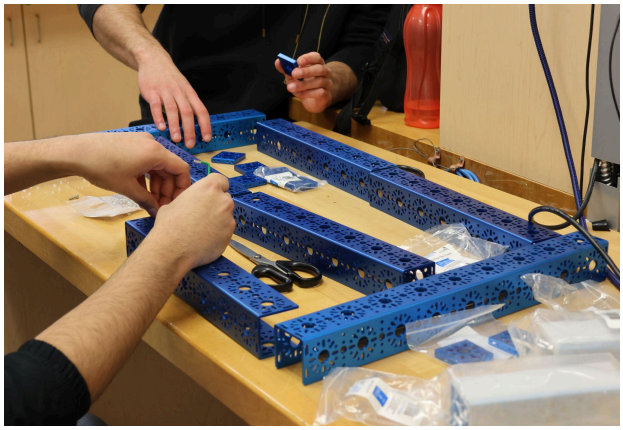
The BuzzRover is powered by four Maverick 12V DC gear motors, chosen for their high torque output at low speeds, which is ideal for moving heavy loads safely. Gear motors are commonly used in agricultural robotics because they provide controlled movement without requiring high speeds, reducing the risk of tipping or

frame damage (ISL Products). Motor control is handled by a Titan Quad Motor Controller, which allows independent control of all four motors. This supports skid-steer operation, balanced power distribution, and smoother turning. Using a single multi-channel motor controller also reduces wiring complexity and improves system reliability.



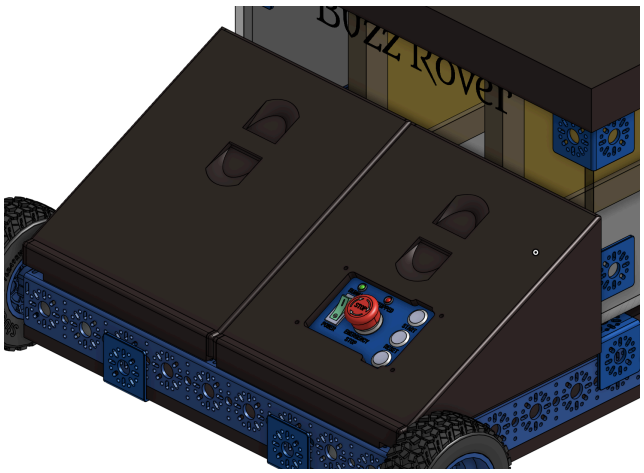
STRUCTURAL FRAME AND ENCLOSURE

The chassis of BuzzRover is constructed using aluminum U-channels, flat beams, and L-beams, fastened with M3 socket head cap screws and Nyloc nuts. Aluminum was the selected material for the rover. Aluminum has a protective oxide layer that prevents it from being corroded or damaged by UV radiation (Wakelin, 2025). It is also lightweight and durable, making it the perfect material to carry heavy honey frames and prevent natural degradation, such as corrosion. Its ease of machining and modular assembly allows the frame to be easily repaired or reconfigured if components need to be replaced or upgraded. Nyloc nuts are used throughout the structure to prevent fasteners from loosening due to vibration during rover operation in the farm. Nyloc nuts also maintain their locking security over a wide temperature range, crucial in an outdoor farm environment over different seasons (“NutsandBolts”, 2026).



FRONT HOODS

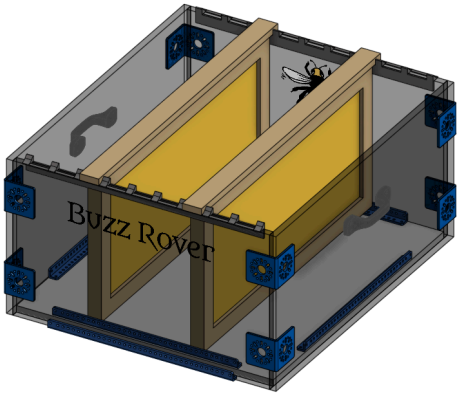
In addition to the aluminum frame, BuzzRover incorporates a student-created custom 3D-printed front hood as an electronic enclosure, which is printed using ASA filament. ASA filament is a material that can withstand high-force shocks and can maintain its structure. ASA is also resistant to chemical and weather factors such as radiation and temperature fluctuations (Bonnin, 2023). A full-length plywood panel mounted underneath the chassis increases rigidity and protects internal components from ground debris and impact. A second plywood panel mounted above the chassis acts as a stable load-bearing surface for the honey frame containment system. This enclosure serves two main purposes. First, it protects internal wiring and electronic components from dust, debris, and accidental contact during operation and demonstrations. Exposed wiring in mobile robots increases the risk of disconnection, damage, or short circuits, particularly in outdoor environments. The enclosure helps reduce these risks by physically shielding sensitive components.



Second, the 3D-printed body improves the overall appearance and professionalism of the rover. A clean, dual-lid modular design makes the system easier to understand, safer to interact with, and more suitable for public demonstration. The enclosure also allows internal components to be organized more effectively, improving maintenance and troubleshooting. The left lid houses the VMX Pi and battery, while the right lid contains the motor controller and integrated control panel. Using 3D printing enabled rapid iteration of the body design and allowed the enclosure to be tailored precisely to the rover's frame and electronics without adding unnecessary weight.

PLEXIGLASS CONTAINMENT SYSTEM

The honey frame containment system is constructed from clear plexiglass panels. Plexiglass was chosen because it is lightweight, impact-resistant, and corrosion-resistant (Acme Plastics, n.d.), making it suitable for agricultural environments while adding minimal weight to the rover. The material allows farmers to visually monitor the honey frames during transport without opening the enclosure, improving safety and operational awareness, and allowing the user to ensure there are no bees entrapped within the containment system. The panels are secured using aluminum L-beams, L-brackets, and M3 fasteners to create a rigid and durable structure. Precise drilling and alignment ensured that the panels fit securely together while maintaining structural integrity. The completed plexiglass box slides into integrated aluminum side walls built into the chassis. These side walls act as guide rails, securing the box in position and preventing lateral movement during operation. This design protects the honey frames from shifting, external contact, and environmental exposure while maintaining a clean, professional, and functional appearance.



CONTROL SYSTEM AND POWER

BuzzRover uses a VMX Robotics Controller, which supports remote operation and allows integration with a handheld controller. A remote control enables the operator to remain at a safe distance from both the rover and the beehives during operation. Electric battery power was chosen to eliminate exhaust emissions, reduce noise, and have a lower operating cost than gasoline (“Battery electric vehicles: The future of clean, efficient transportation”, 2025). These advantages align with sustainable agricultural practices.

LOAD MANAGEMENT AND MODULARITY

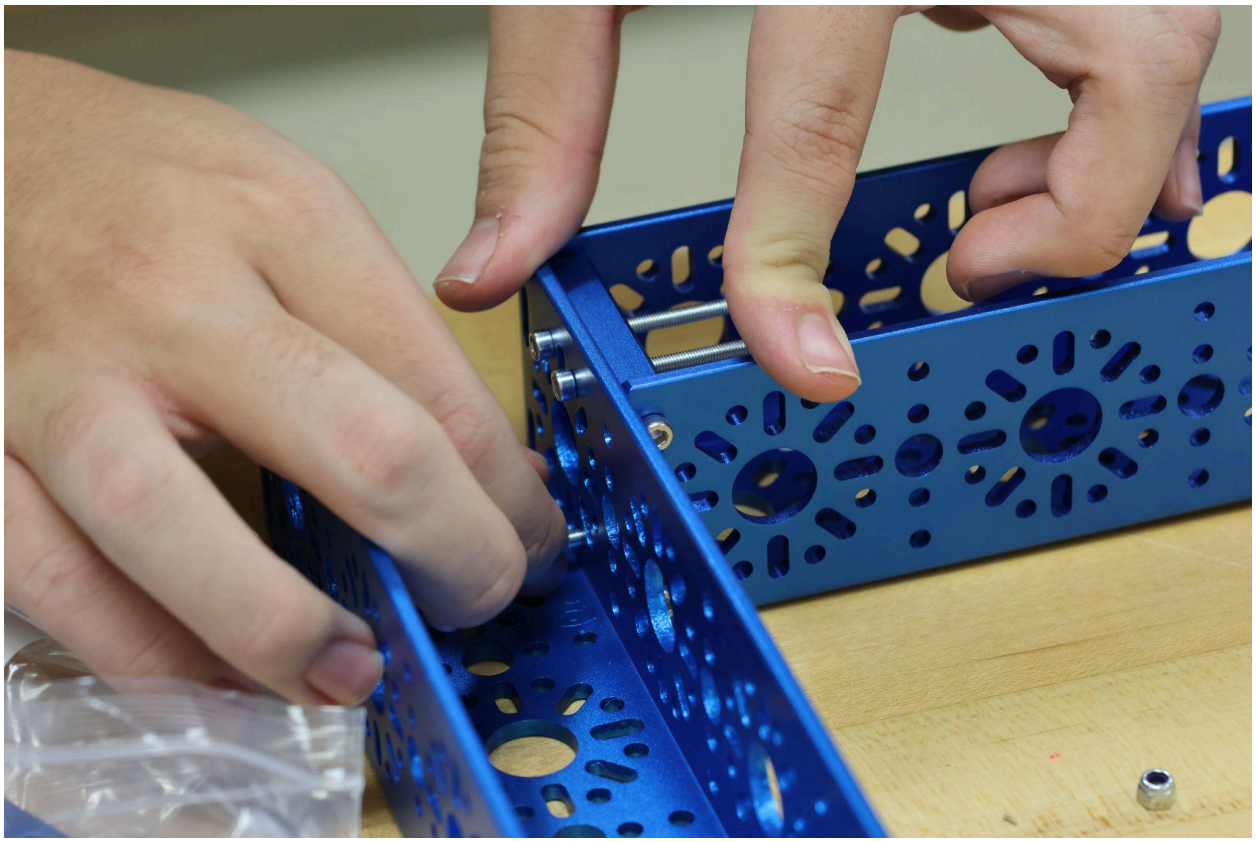
The inclusion of a 9-frame spacer ensures proper support and alignment of honey frames during transport, minimizing the risk of tipping or damage. The frame-based construction allows for future modular additions, such as expanded racks or lifting mechanisms, without redesigning the entire rover.

WHY THIS DESIGN WAS CHOSEN

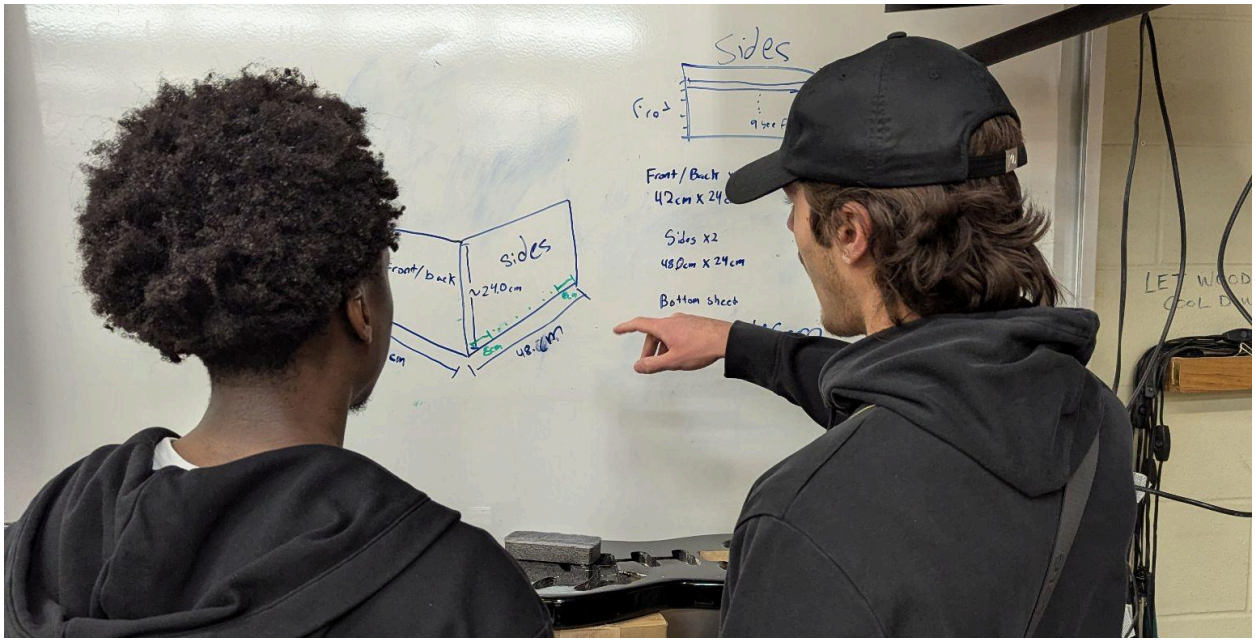
Each design decision for BuzzRover was made to balance strength, stability, simplicity, and safety. The rubber wheel drivetrain improves performance on uneven terrain, the aluminum structure provides durability without excessive weight, and the four-motor skid-steer system enables precise control. Together, these choices create a rover that is practical, scalable, and well-suited to real agricultural use.

DESIGN PROCESS

Our design process for the rover evolved through iterative testing and hands-on refinement. We began by constructing a modular aluminum chassis using aluminum U-beams connected with L flat brackets and end-piece plates. Once the frame was assembled, we installed the DC gear motors and wheels to validate our control code before moving forward with enclosure and load integration. Early testing also ensured that our power distribution and skid-steer logic functioned correctly under basic conditions.



We then moved on to the honey frame containment system. Our original design included welded aluminum L-beams to permanently secure the plexiglass panels. However, we quickly recognized that a fixed enclosure would make cleaning difficult. Since honey residue and debris are common in agricultural environments, we redesigned the system to be removable. The updated design allows the plexiglass box to slide into aluminum side walls, creating a secure but detachable containment system that improves maintenance and long-term usability.



While working on the plexiglass assembly, we encountered challenges with the plexiglass material itself, as aged protective film had bonded to the surface. Removing this residue required careful mechanical scraping and the application of hot water and alcohol to restore the panels without causing structural or cosmetic damage. During assembly, we experienced several measurement inconsistencies. Because certain parts were marked, disassembled, drilled, and later reassembled, exact positioning was difficult to replicate. This led to minor hole misalignments that required careful re-drilling and realignment using a hand drill. Through this experience, we significantly improved our marking and clamping techniques to minimize errors in future iterations.

Simultaneously, we began 3D printing the front hood enclosure. Our original front hood design was too large to print as a single piece and limited access to internal electronics. To address both fabrication and accessibility issues, we redesigned the enclosure into a dual-lid modular system. Each lid was printed in two components, a base and a cover. We selected ASA (Acrylonitrile Styrene Acrylate) as the printing material due to its high heat resistance, mechanical strength, and UV stability, which make it well-suited for outdoor agricultural environments where elevated temperatures and sun exposure are expected (Khan, 2026). This solution not only resolved printing constraints but also improved

maintenance access to the battery, VMX Pi, motor controller, master switch, and emergency stop.

To further enhance field durability, we incorporated a waterproof enclosure strategy into the hood design. The lid interfaces were designed with overlapping lips to reduce direct water ingress, and the mating surfaces allow for gasket integration to improve sealing performance. Additionally, we implemented a vertical sliding rail mechanism that enables the lid to move smoothly up and down along guided tracks. This rail system allows the enclosure to be separated vertically without lateral shifting, reducing stress on internal wiring and connectors during removal. A reinforced integrated handle was added to the top section to facilitate controlled lifting and removal of the lid. Together, the sliding rail and handle system provide efficient access for maintenance while maintaining environmental protection and structural stability during operation. The base sections printed successfully, but the tall cover sections failed multiple times due to the height and the warping of the ASA. However, through careful reconfiguration of the printer, we were able to print it successfully.

After completing the enclosure redesign, we moved on to installing the plywood reinforcement panels which we painted black. We began with the bottom plywood panel. During installation, we realized we could not secure bolts with nuts on the reverse side of the chassis due to limited access space. Instead of disassembling major structural components, we implemented self-tapping screws combined with aluminum L-brackets. The L-brackets were fastened directly to the chassis, and the plywood was secured to the brackets, effectively locking it into place. This approach maintained structural rigidity while working within our spatial constraints. The same fastening method was used for both sides of the bottom panel. We then installed the upper plywood panel using a similar bracket-supported system to ensure stability and load support for the plexiglass containment system.

Our original plan was to glue the aluminum side walls directly to the plywood platform to guide the plexiglass box. However, there was insufficient clearance space, and permanently bonding the walls would reduce modularity and limit

future adjustments. Instead, we repositioned the walls further outward and mounted them directly to the aluminum chassis using self-tapping screws. This approach maintained structural stability while keeping the plexiglass system removable and properly aligned.

Overall, our design process was iterative and problem-driven. Fabrication limits, measurement inconsistencies, and mounting constraints required multiple adjustments to our original concepts. By redesigning components, modifying fastening methods, and improving modularity, we transformed early limitations into functional improvements. The final system reflects stronger structural integrity, better accessibility, and greater long-term serviceability for real agricultural use.

SAFETY

Safety was a primary consideration throughout the design, construction, and operation planning of BuzzRover. The rover is designed to comply with the AgRobotics Student Challenge safety requirements and to operate safely within an agricultural demonstration environment.

ELECTRICAL SAFETY

BuzzRover is powered by a single 12V sealed battery supplied as part of the Studica robotics kit, in accordance with competition rules. The battery is securely mounted to the chassis to prevent movement during operation.

All electrical wiring is routed along the bottom of the frame and secured to reduce strain, abrasion, or accidental disconnection. Electrical connections are insulated to prevent exposed conductors. Wiring is selected to be appropriately rated for the expected current draw of the drivetrain and control electronics.

A main electrical protection device provided with the Studica kit is used to protect the system from overcurrent conditions. Power to the rover can be shut off immediately when required. All electrical terminals remain insulated during operation.

MECHANICAL SAFETY

BuzzRover uses pneumatic rubber wheels/treads from the Studica kit, which provide traction while minimizing damage to the dirt field and surrounding surfaces. The chassis is constructed from aluminum framing, providing a rigid structure that resists bending and warping.

Rotating components such as wheels and drivetrain elements are positioned to minimize exposure. Potential pinch points are reduced through component placement, guarding where appropriate, and controlled operating speeds. The rover is designed with a low center of gravity, reducing the risk of tipping when transporting heavy honey supers. The overall system weight remains well under the 150 lb limit, and the rover is fully self-supported during demonstrations.

OPERATIONAL SAFETY

BuzzRover is controlled using a remote control system, allowing the operator to remain outside the demonstration area at all times. The rover is programmed for smooth acceleration, controlled turning, and gradual stopping to reduce instability and mechanical stress.

All testing and demonstrations are conducted within the designated 12’ × 8’ dirt field or approved surrounding concrete area. The rover is operated at low, controlled speeds appropriate for agricultural use.

PERSONAL SAFETY PRACTICES

Team members follow standard workshop and event safety procedures, including:

- Wearing CSA-approved safety glasses during construction, testing, and demonstrations
- Wearing closed-toe footwear
- Securing loose clothing, hair, and jewelry
- Using tools only after receiving proper instruction and supervision

These measures ensure BuzzRover can be demonstrated safely and responsibly in a public agricultural robotics setting.

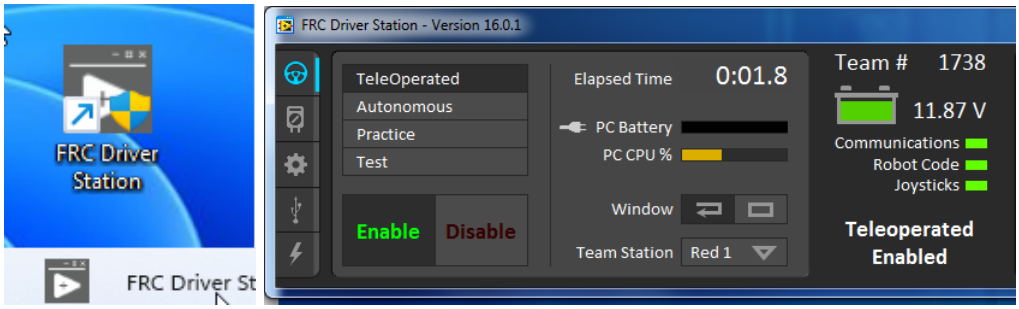
USER INSTRUCTIONS

REMOTE OPERATION

1. Ensure the *BuzzRover* has a charged battery.
2. Toggle the green “on” button, and ensure the emergency stop button isn’t pushed in.

   When the robot is running, the green LED is turned on, and when the emergency stop is engaged, the red LED is turned on.

3. To connect your laptop device to the *BuzzRover*, go to WiFi settings and connect to the local network “*WorldSkills-1234*” with the password “*password*”.
4. Go to the *FRC Drive Station* app and select *Start TeleOp*.



5. Connect your controller to the laptop device via a USB port.

6. Tilt the left joystick up and down to move the *BuzzRover* forwards and backwards, and tilt the joystick left and right to rotate.

PATH RECORDING

1. Press the bottom button ✕ to start recording a path.

2. Move the robot around to store your recording, and press the bottom button ✕ again to stop recording.
3. To replay your recording, press the top button △.
4. Use the bumpers L and R to scroll through which recording you are writing to or reading from, which also displays the time it takes to complete the path.

    There are 5 recording streams total, which will be displayed on your laptop device.

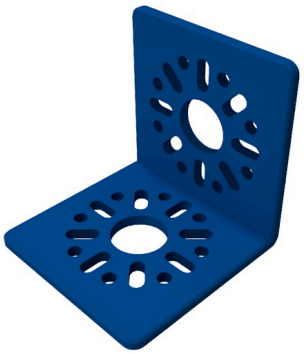
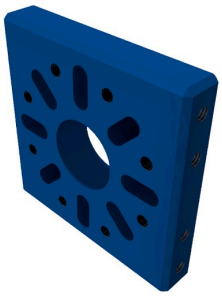
BATTERY REPLACEMENT AND MAINTENANCE

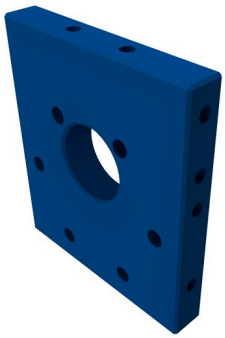
1. To replace a battery, open the left hood on the front of the robot.
2. Disconnect the dead battery by detaching the red and black cables.
3. Insert your new battery into the orange case and connect the wires of the same color.
4. Ensure all wires are connected within the hood.
5. Replace the hood by lowering it vertically onto the robot.

PARTS LIST

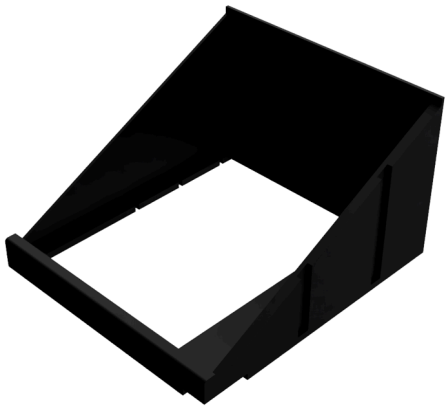
BuzzRover			
Base			
Item #	**Name**	**Quantity**	**Picture**
0001	M3 x 10mm Socket Head Cap Screw	24	
0002	M3 x 12mm Socket Head Cap Screw	16	
0003	8 in x ½ in Self-Tapping Screw	20	
0004	M3 Nyloc Nut	40	

0005	144mm U-Channel	4	
0006	432mm U-Channel	2	
0007	336mm U-Channel	4	

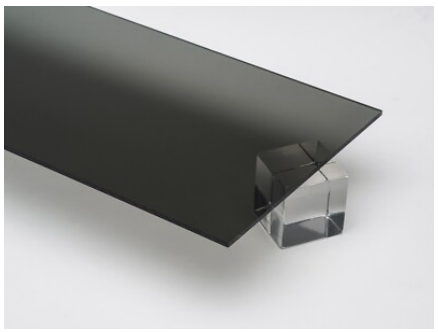
0008	288mm L Beam	6	
0009	96mm x 40mm Flat Bracket	6	
0010	48mm x 48mm L Bracket	10	
0011	End Piece Plate	14	

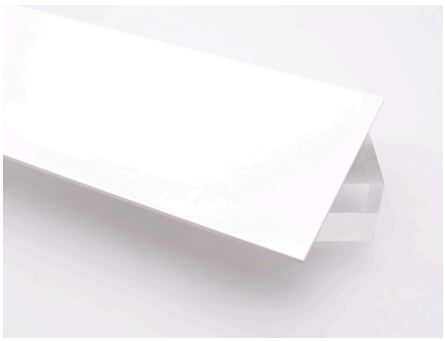
0012	Maverick 12V DC Gear Motor	4	
0013	Clamping Hub 6mm D Shaft - v2	4	
0014	Motor Mount Plate	4	
0015	76.8cm x 43.2cm Cut Plywood (Painted Black)	1	

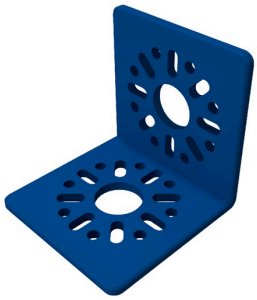
0016	52.8cm x 43.2cm Cut Plywood (Painted Black)	1	
0017	125mm All-Terrain Wheel Set	1 set, 4 wheels	
3D-Printed Hood			
0018	Hood Left Base ASA Filament - 256g	1	

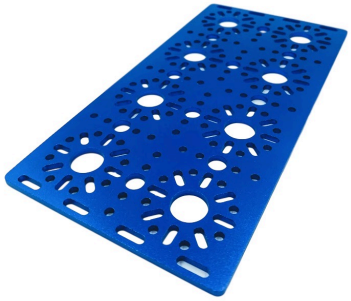
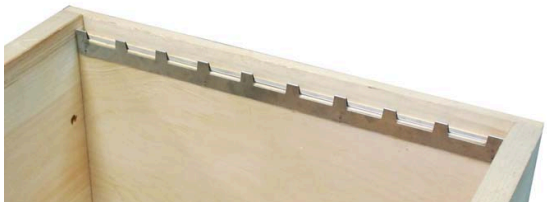
0019	Hood Right Base ASA Filament - 256g	1	
0020	Hood Left Lid ASA Filament - 312g	1	
0021	Hood Right Lid ASA Filament - 293g	1	

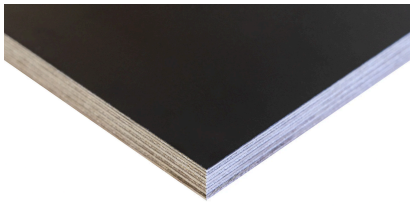
0022	Control Panel Module	1	
0023	M3 x 25mm Socket Head Cap Screw	4	
0024	8 in x ½ in Self-Tapping Screw	4	
0025	Titan Quad Motor Controller	1	
0026	VMX Robotics Controller	1	

0027	Battery Pack Charger	1	
0028	Battery Pack	1	
Containment System			
0029	0.5 in Clear Cut Acrylic Plexiglass Sheet Cut to 48cm x 24cm	2	
0030	0.5 in Tinted Cut Acrylic Plexiglass Sheet Cut to 42cm x 24cm	2	

0031	0.25 in White Cut Acrylic Plexiglass Sheet, Cut to 41cm x 46cm	1	
0032	M3 x 12mm Socket Head Cap Screw	48	
0033	M3 x 25mm Socket Head Cap Screw	4	
0034	8 in x ½ in Self-Tapping Screw	14	

0035	M3 Nyloc Nut	52	
0036	288mm L Beam	6	
0037	48mm x 48mm L Bracket	8	
0038	96mm x 40mm Flat Bracket	1	

0039	144mm x 40mm Flat Bracket	4	
0040	192mm x 96mm Flat Bracket	4	
0041	9 Deluxe Metal Frame Spacer	2	

0042	Handles ASA Filament - 180g (per handle)	2	
Lid			
0043	52.8cm x 46.3cm Cut Plywood (Painted Black)	1	
0044	52.8cm x 4.2cm Black Cut Plywood (Painted Black)	2	
0045	46.3cm x 4.2cm Cut Plywood (Painted Black)	2	

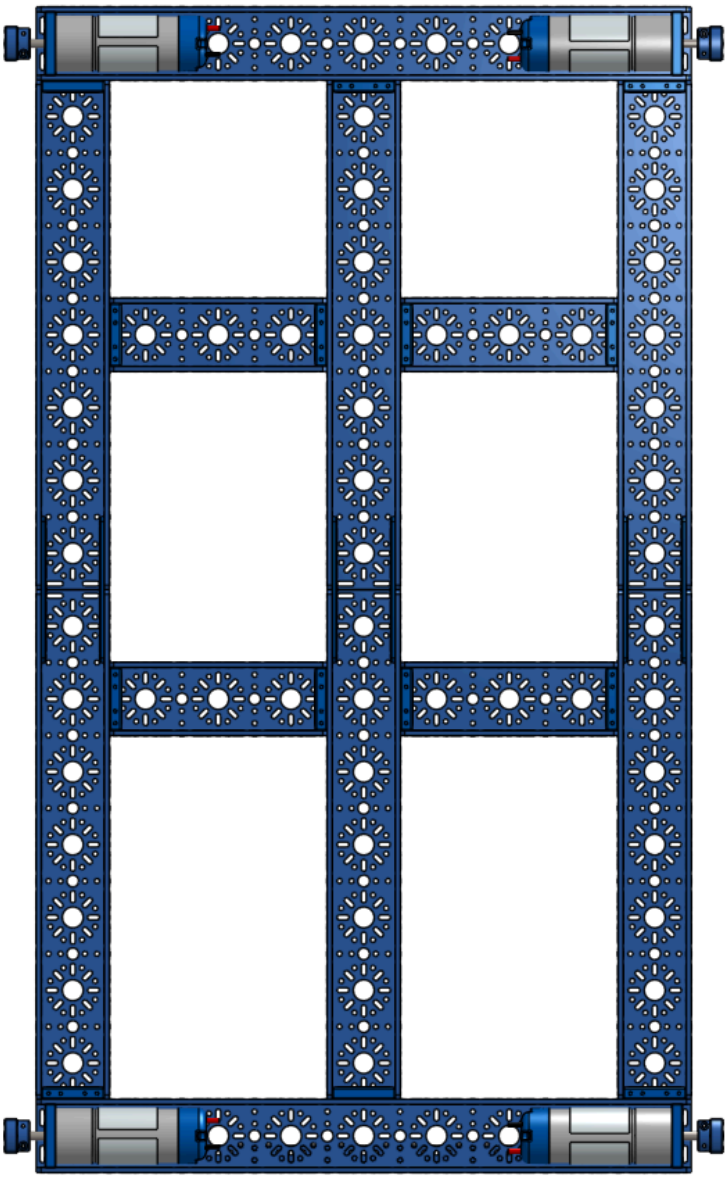
0046	Handles ASA Filament - 180g (per handle)	2	

CAD DIAGRAMS

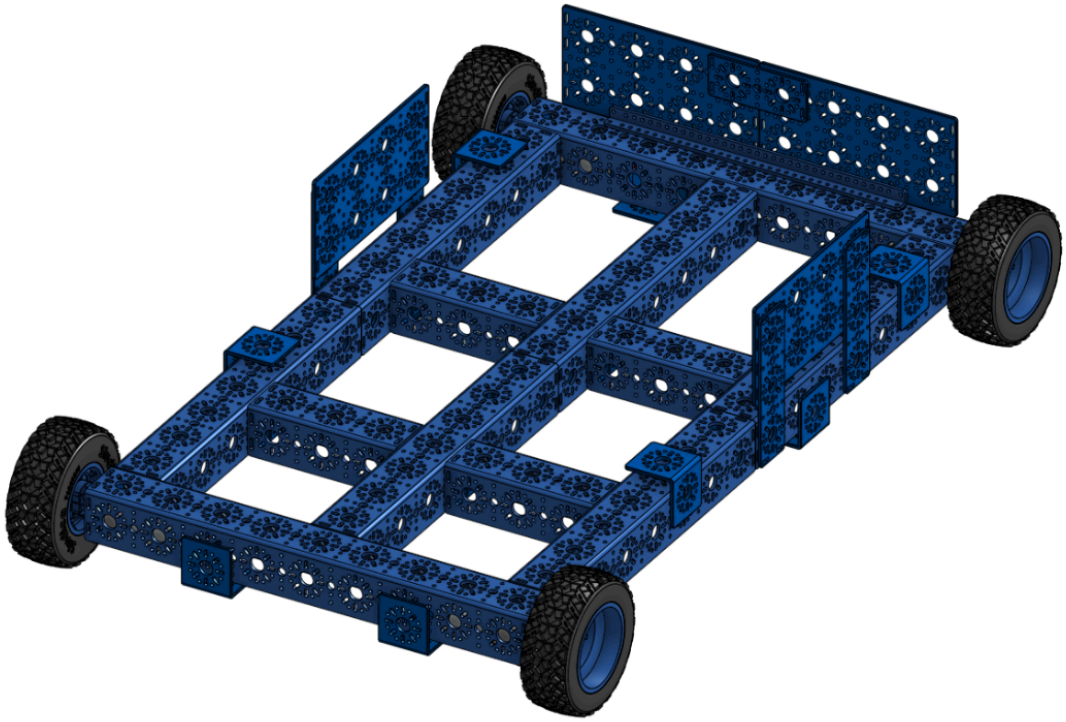
ONSHAPE CAD REPOSITORY LINK

[Onshape Model For Rover](#) [Mhanna & Tailor]

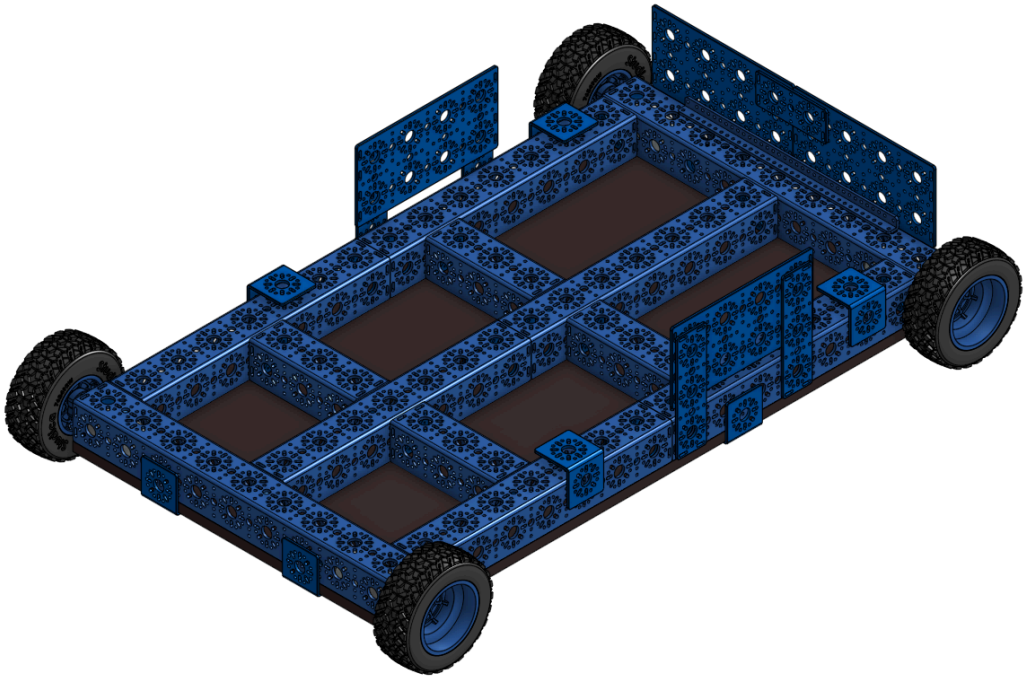
Chassis Base (Bottom View):



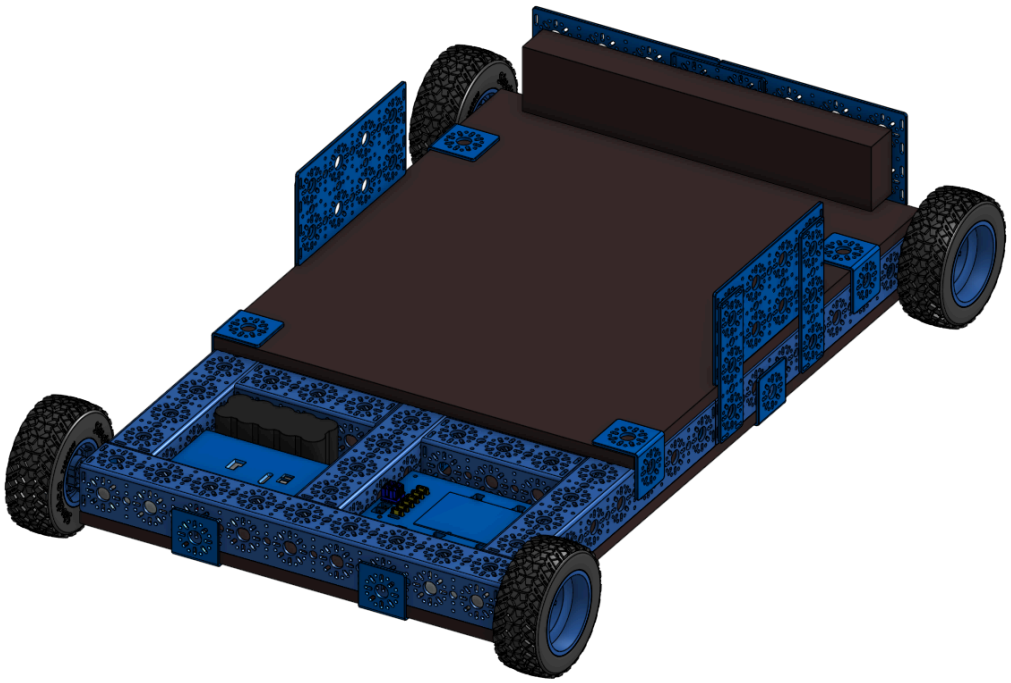
Drivetrain Assembly:



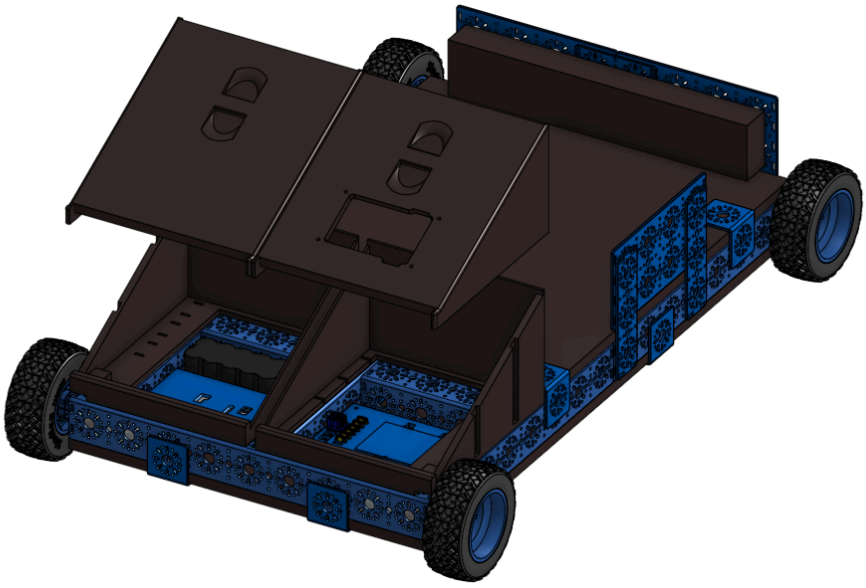
Lower Plywood Configuration:



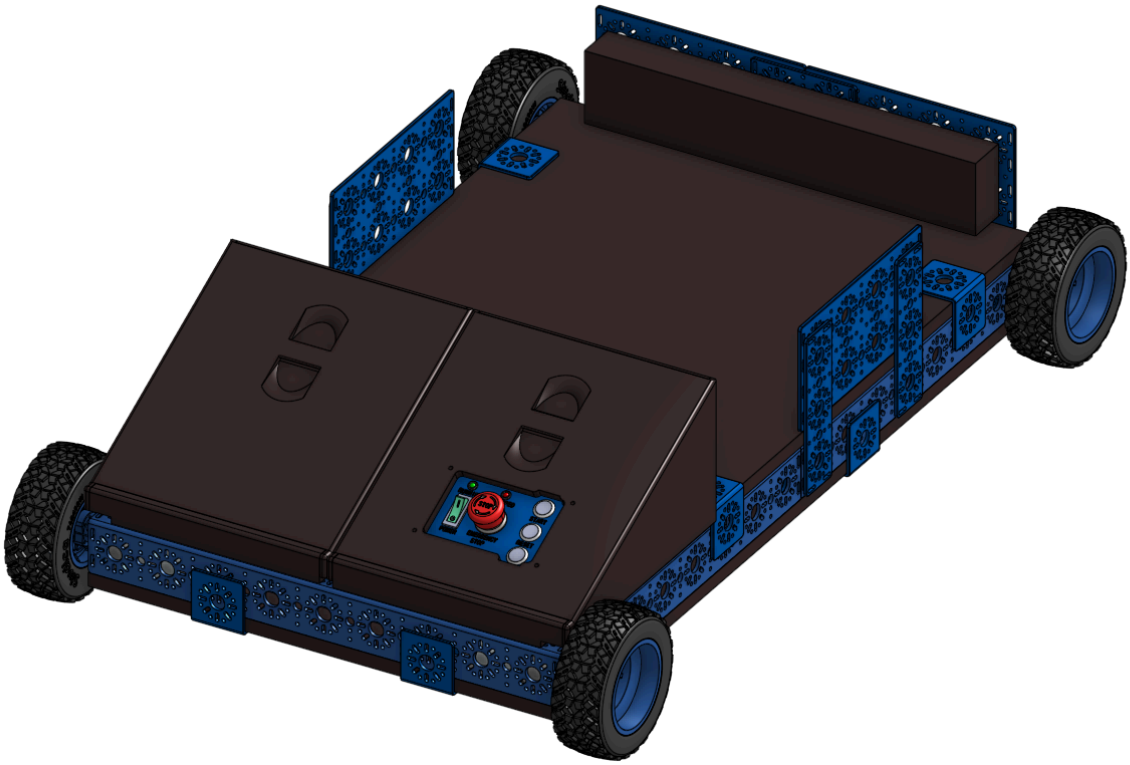
Upper Plywood & Electronic Bay:



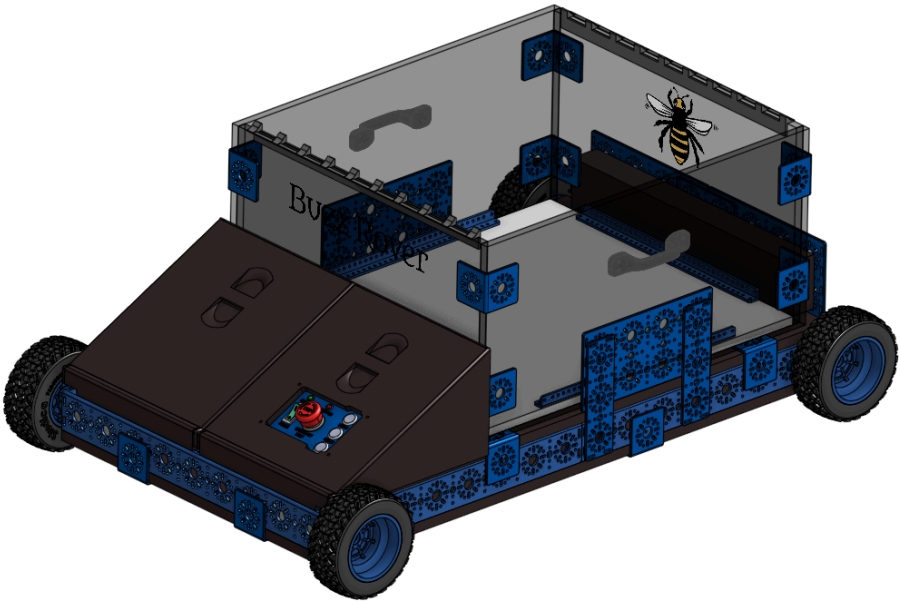
Access Hoods:



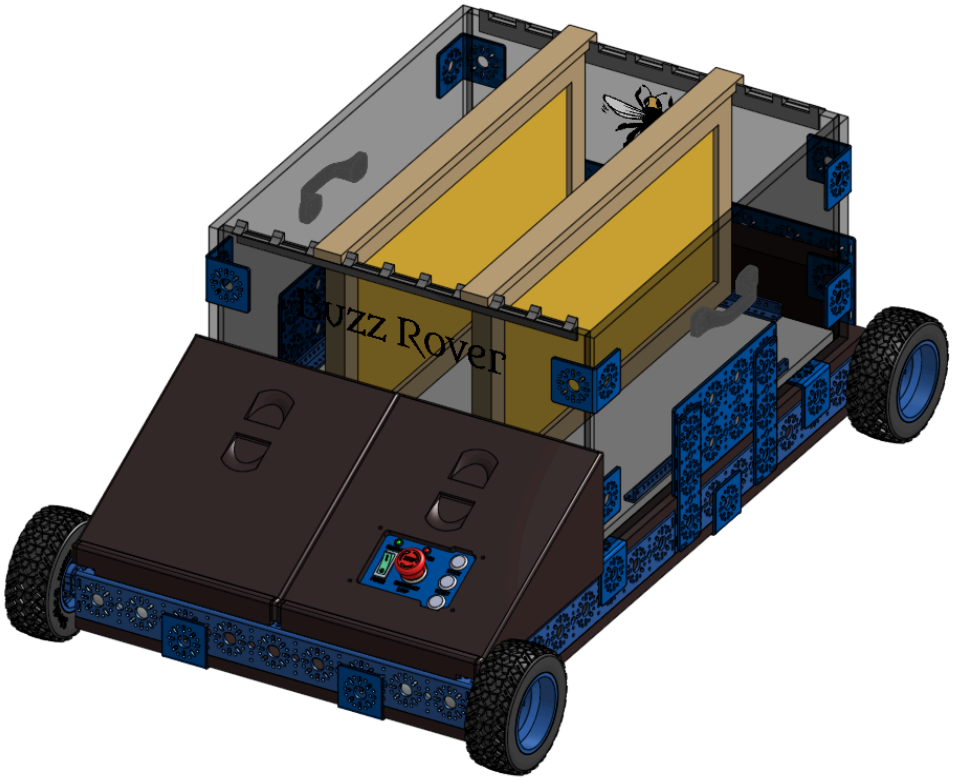
Control Interface:



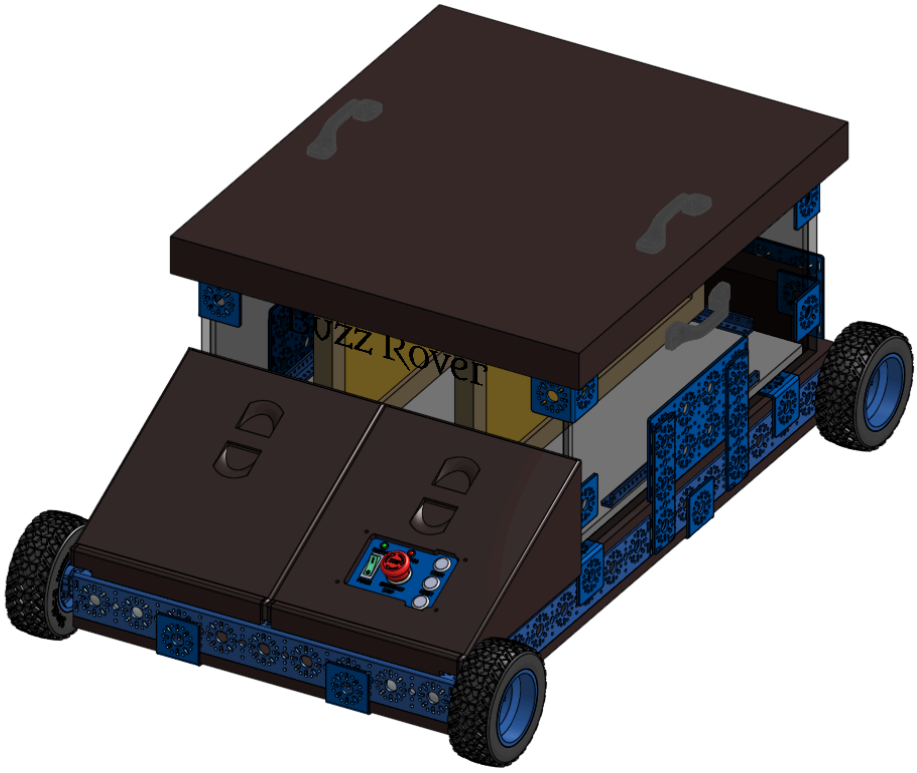
Plexiglass Enclosure:



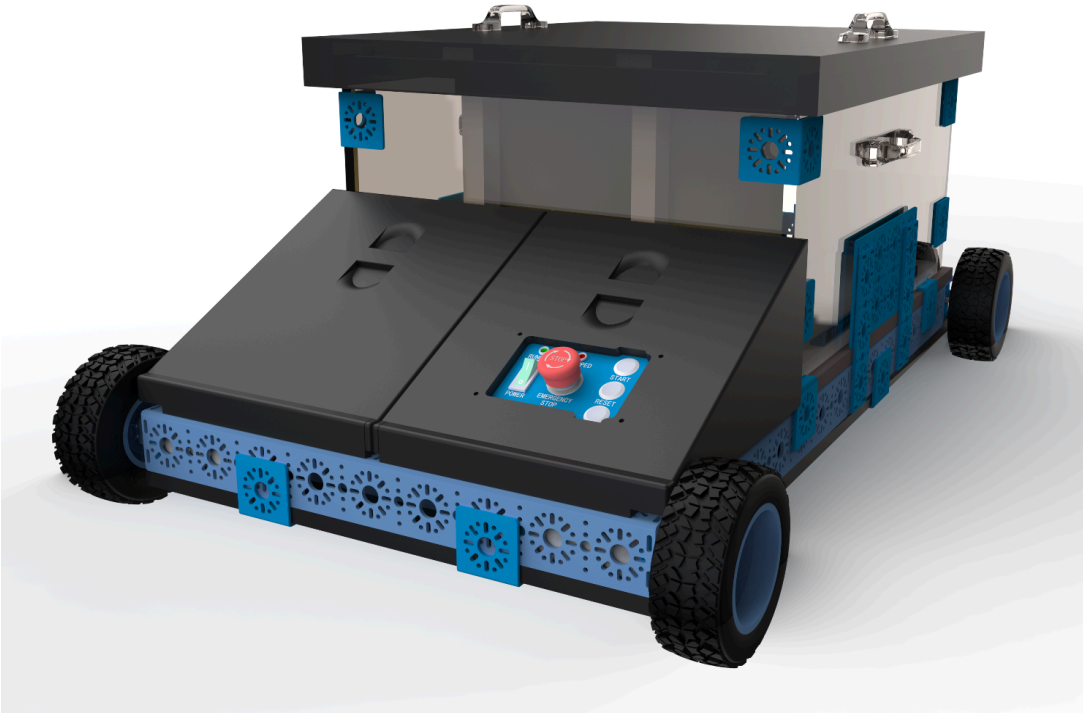
Plexiglass Enclosure with Honey Frames:



Complete Model:

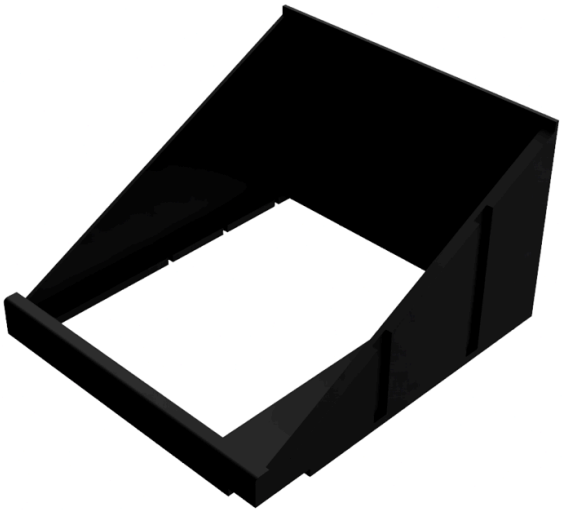


Realistic Rendering:

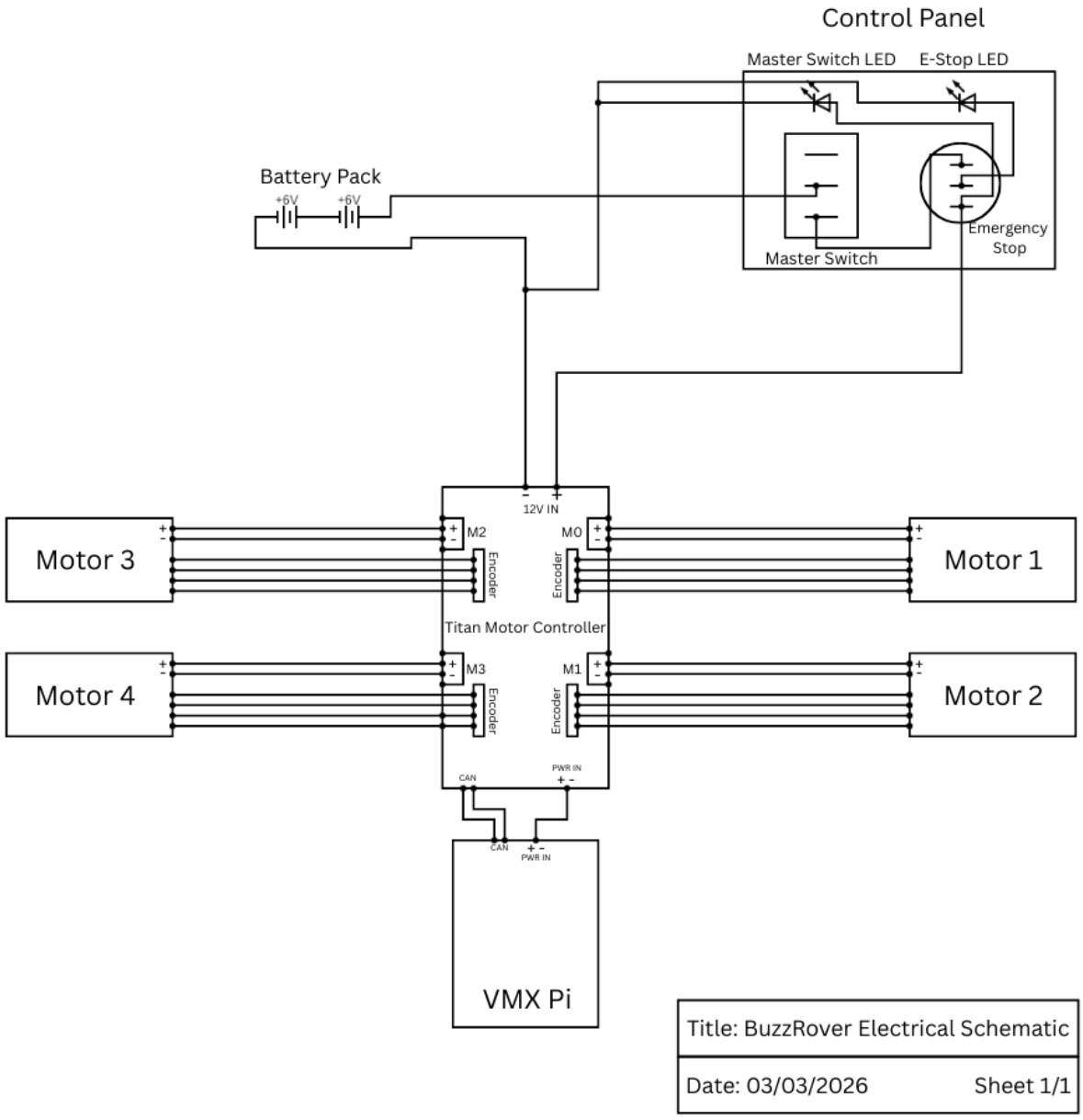


[Onshape Model For Hood and Handle](#) [Chen][Sidrake]

[Heavy Duty Handle Source](#)[Filopat]



ELECTRICAL DIAGRAM



CODE

GITHUB SOURCE CODE LINK

Code is currently posted in a public repository [Peacock].

OVERVIEW

This section provides a high-level description of the software features of the rover and how they are implemented.

Remote control is provided via the `DriveTrain`, `OI` and `Teleop` classes. `DriveTrain` handles converting driving commands into motor control signals. `OI` (Operator Input) provides an interface to access controller inputs. Finally, `Teleop` ties them together by reading controller input from `OI` and sending commands to `DriveTrain`.

Path recording is mainly handled by the `PathRecorder` class, and the `Sample` class. When recording a path, the `PathRecorder` class saves samples of the motor speeds every 20ms into one of the eight arrays of `Sample` objects (path slots). Then, when replaying a path, `PathRecorder` reads a sample, and sends a command to `DriveTrain` to set the motor speeds. This process repeats every 20 milliseconds, ensuring that the path replay is time-accurate.
`TeleOp` also handles informing `PathRecorder` when the operator changes the current path slot, or when starting to record or replay a path. It also interprets the data of the motor encoders to implement stall detection, ensuring no battery is wasted.

Paths are saved to disk as JSON files so they persist between power cycles. When a recording is stopped, PathRecorder saves it to `/home/lvuser/paths/` as a JSON file such as `path_1.json`. When the robot starts up, `PathRecorder` loads all saved paths from that directory back into memory.

Key code snippets are available in Appendix A

DETAILED OUTLINE

This section aims to explain the code, function by function and class by class.

(RobotContainer.java) RobotContainer() is the init function for the class, and simply initializes the DriveTrain and OI subsystems. This is run at the very beginning, and then it sets the default command to TeleOp.

(Constants.java) contains a variety of constants. These include the CAN ID of the Titan, being 42 always, the motor IDs, which will always be 0 to 3, the deadzone of the joystick, and constants relating to the encoders.

(DriveTrain.java) DriveTrain() is the init function for the class. It sets up the drive train for the robot, by defining each motor, their direction, and their encoders.

(DriveTrain.java) driveArcade() drives the robot in arcade mode. It takes in the two values, the X and Y variables, clamps them between -1 and 1, then calculates how much strength to push the motors with.

(DriveTrain.java) driveTank() drives the robot in tank mode. It takes in the two values, the X and Y variables, clamps them between -1 and 1, then calculates how much strength to push the motors with.

(DriveTrain.java) resetGyro() does exactly that: resets the gyro, or the yaw.

(DriveTrain.java) resetEncoders() resets the encoder distance for each motor. The motors keep track of how far they have rotated, and we can reset this variable by setting it to 0.

(DriveTrain.java) getAverageEncoderDistance() does exactly what it says on the tin: takes the average of all 4 of the motors' encoder distances.

(DriveTrain.java) leftBackEncoderDistance() takes the left back motor's encoder distance.

(DriveTrain.java) rightBackEncoderDistance() takes the right back motor's encoder distance.

(DriveTrain.java) leftFrontEncoderDistance() takes the left front motor's encoder distance.

(DriveTrain.java) rightFrontEncoderDistance() takes the right front motor's encoder distance.

(DriveTrain.java) getLeftEncoderDistance() returns the average encoder distance of the left motors.

(DriveTrain.java) getRightEncoderDistance() returns the average encoder distance of the right motors.

(DriveTrain.java) toggleRecording() toggles the path recording. (See Appendix A)

(DriveTrain.java) samplePath() records a path sample.

(DriveTrain.java) startReplay() begins the replay of a specific segment of the recorded path.

(DriveTrain.java) stopReplay() stops the path replay and sets all the values of the motors to 0, and stops driving.

(DriveTrain.java) replayStep() executes one step of the path replay. It gets the motor powers from the path recorder and drives the robot with them. Returns false when the replay is finished. (See Appendix A)

(DriveTrain.java) nextPath() cycles to the next path slot and prints which path you switched to.

(DriveTrain.java) prevPath() cycles to the previous path slot and prints which path you switched to.

(DriveTrain.java) isRecording() returns whether the path recorder is currently recording.

(DriveTrain.java) isReplaying() — returns whether a path is currently being replayed.

(OI.java) is the operator input wrapper. It reads the controller and calculates deadzones for joysticks and gets the POV.

(GamepadConstants.java) contains another variety of constants, each relating to an ID for a button or control on the gamepad (controller).

(TeleOp.java) Teleop() is the init function, and just states that Teleop requires the DriveTrain subsystem.

(TeleOp.java) initialize() resets the encoders and resets the yaw (rotation) variable.

(TeleOp.java) execute() is the main loop. It takes the left stick measurements as read by OI.java and runs driveArcade() from driveTrain.java to move the robot as well as the bumpers to cycle path slots, and X and Y for recording & replaying. (See Appendix A)

(TeleOp.java) end() sets all the values of the motors to 0, and stops driving.

(PathRecorder.java) Sample is a sample of a path, in the form of an object. Samples are recorded many times a second, and each sample stores an average of each motor's velocities over the sample time. When replaying, these samples act as a linked list, and so when replaying the samples, the first object sets the motor velocities to what was stored for its given time, and then moves on to the next sample.

(PathRecorder.java) PathRecorder() edits the path array to store the objects.

(PathRecorder.java) currentSamples() returns the samples at the current path index.

(PathRecorder.java) startReplay() initializes replay from the beginning of the recorded path segments

(PathRecorder.java) startRecording() clears any old recorded data, stores the starting encoder positions, and begins recording a new path.

(PathRecorder.java) stopRecording() flushes any remaining accumulated data as a final segment, then stops recording.

(PathRecorder.java) sample() takes in the current encoder distances and motor powers. It accumulates the encoder deltas and motor powers each cycle, and commits a segment when enough distance is covered.

(PathRecorder.java) startReplay() initializes replay from the beginning of the recorded path segments.

(PathRecorder.java) replayStep() calculates how much power to push the motors with for the current segment, based on how far the robot has traveled. It advances at most one segment per call, and ramps down the power as it gets close to finishing a segment. Uses motor encoder values to ensure the robot is not stalled. Returns null when the replay is finished. (See Appendix A)

(PathRecorder.java) getSegmentCount() does exactly what it says on the tin: returns the number of recorded segments.

(PathRecorder.java) nextPath() cycles to the next path slot.

(PathRecorder.java) prevPath() cycles to the previous path slot.

(PathRecorder.java) getCurrentPathIndex() returns the current path slot index.

(PathRecorder.java) checkStall() checks if the robot is stalled during replay by comparing encoder values over multiple cycles. Only triggers when the robot is actually being told to move.

(PathRecorder.java) getPathDuration() returns the duration of the current path in seconds.

(PathRecorder.java) savePath() saves a single path slot to a JSON file on disk.

(PathRecorder.java) loadPath() loads a single path slot from a JSON file on disk.

(PathRecorder.java) loadAllPaths() loads all path slots from disk on startup so paths persist between power cycles.

PARTNER REFLECTION

Our community partner, Karin's feedback is very encouraging and insightful. She described the project as "user friendly" and "very usable for beekeepers," which reinforced that our work has practical value. She also mentioned that the Honey Bee Research Centre in Guelph might be interested, which highlighted potential real-world applications. However, she suggested that we should make the rover a bit taller and add a lid, which are improvements we implemented after the interview. Overall, her comments helped us recognize both the strengths of our project and the areas where small adjustments could further improve its usability and impact it could make on agriculture.

STUDENT REFLECTION

This project was our first major group-focused initiative, and rooted in a good cause at that. Unlike isolated classroom builds or personal projects, BuzzRover required us to think beyond simply “making it work”, since we were entrusted with designing something reliable, safe, durable, and practical for Karin’s specific agricultural application. The entire creative process was an enormous undertaking, providing everyone with a valuable learning experience and lessons learned. Whether it was the design team, physical build team, or software team, there was the presence of the inevitable brick wall, which we thought we could not overcome.

However, in many ways, this experience represented the true essence of the engineering design process: encountering a problem, brainstorming creative solutions, choosing a design, testing, and optimizing it. Having said so, the CAD design, prototyping, fabrication, or system integration processes all proved to have unique challenges that required highly creative thinking. Designing in a physical environment is radically different from a digital one, where measurements are perfect, clearances never interfere, and components align exactly as intended, so everything appears flawless. In actuality, it could not be further from the truth.

Materials will flex, drill bits will wander, tolerances will stack, and you will run out of nuts. However, as a team, we knew that giving up would mean abandoning our community partner, and this was not a viable option. We knew that confronting and circumventing these challenges would not be a trivial matter, so let's explore how we delved into the possibilities of solutions: (1) The physical prototyping enlightened us to the instrumental role of proper, sustainable preparation. Making reasonable and insightful decisions, and drafting projections and time estimates for what might take the most time, or treating some physical components prudently would have significantly facilitated the build process.

For all of us, this was our first instance working with acrylic sheets, and the build team hadn't anticipated the impending wave of obstacles presented by poorly aged filament, resulting in a deeply protracted treatment process; however the job was completed. The aforementioned, along with several other instances of trouble, provided us with invaluable insights into a typical build process, which we can take with us to apply across anything; (2) The software and digital control team truly had to exercise the greatest form of patience and resilience, experiencing incredibly demoralizing errors and willpower-eroding setbacks. The project's digital environment broke down on several occasions, and we were forced to go so far as to change the hardware we were using to render the program functional; (3) The culmination of our learnings would most likely be attributed to the fact that engineering is not defined by immediate success, but by the consistent tackling of the problem and analysis from all angles. Thus, the power of iteration and moving past our "brick walls" served as checkpoints that demanded creativity, collaboration, and the unyielding nature of our group efforts. Ultimately, we are grateful for each of the obstacles we encountered, as they brought us closer to a practical, long-term solution for an ever-present issue.

REFERENCES

Johnson. (n.d.). *Title: Why NiMH batteries are ideal for heavy-duty equipment,: Johnson Eletek*. https://www.zscells.com/news/why-nimh-batteries-are-ideal-for-heavy-duty-equipment/

AcmePlastics. (2019). What is Acrylic / Plexiglass? | Acme Plastics. Acmeplastics.com. https://www.acmeplastics.com/what-is-acrylic-plexiglass

Bonnin, J. (2023, October 20). *Everything you wanted to know about asa filament.* COEX 3D. https://coex3d.com/blogs/blog/everything-you-wanted-to-know-about-asa-filament

Battery electric vehicles: The future of clean, efficient transportation. (2025, January 23). Business & Industry Canada. https://www.industryandbusiness.ca/battery-electric-vehicles-the-future-of-clean-efficient-transportation/

Canada, A. and A.-F. (2020, January 23). *Governments supporting Ontario's beekeeping sector*. Canada.ca. https://www.canada.ca/en/agriculture-agri-food/news/2020/01/governments-supporting-ontarios-beekeeping-sector.html

Canada, A. and A.-F. (2025, December 10). *Government of Canada*. Agriculture and Agri-Food Canada. https://agriculture.canada.ca/en/sector/horticulture/reports/statistical-overview-canadian-honey-and-bee-industry-2024

Chen, B., & Sidrake, R. (2026, March 4). *Hood and Handle*. OnShape. https://cad.onshape.com/documents/813173638ccbcf19d1be87e3/w/771c3e22a91134c99cf0468b/e/3e5430a45d3c69b68f37a0f7?renderMode=0&uiState=69a83a36ef80416872fc3cf0

Filopat. (2026 March 4). *Heavy duty handle*. Thingiverse. https://www.thingiverse.com/thing:130737

ISL Products. “DC Motors and Small DC Gear Motors - the Basics.” ISL Products International, 2025, islproducts.com/design-note/dc-motor-dc-small-gear-motor-basics/.

Khan, M. U. (2026, January 27). *Stop wasting filament: Is ASA or ABS better for your next tough project? All3DP*. https://all3dp.com/2/asa-vs-abs-differences/

NutsandBolts.com. (2026). Nylon lock nuts: Applications, advantages, and compatibility with bolts. https://nutsandbolts.com/pages/nylon-lock-nuts-applications-advantages-and-compatibility-with-bolts

Peacock, J. (2026, March 4). *Agrobotics*. GitHub. https://github.com/ozzyDev27/Agrobotics

Mhanna, J., & Tailor, K. (2026, March 4). *The BuzzRover*. OnShape. https://cad.onshape.com/documents/2498df97e9108e933e170552/w/00a14cf30ac5adb2425ceb92/e/2946553316bce70ce40fa060

Rubber tires: Mastering the artistry of every dynamic ride. Tires. (2025, July 29). https://www.tires-easy.com/blog/rubber-tires

Tracked vs wheeled skid steers: Which is best for my jobsite? https://www.leavittmachinery.com/blog/information/tracked-vs-wheeled-skid-steers-which-is-best-for-my-jobsite

Wakelin, A. (2025, June 1). *Why can aluminium be used outdoors?: Vulcan cladding*. Vulcan Cladding Systems. https://www.vulcansystems.co.uk/blog/why-can-aluminium-be-used-outdoors/

APPENDIX A: SOURCE CODE SNIPPETS

Teleop.java: execute()

```java
public void execute() {

        if (!driveTrain.isRecording() && !driveTrain.isReplaying()) {
            if (oi.getDriveLeftBumper()) {
                driveTrain.prevPath();
            }
            if (oi.getDriveRightBumper()) {
                driveTrain.nextPath();
            }
        }

        if (oi.getDriveXButton()) {
            if (driveTrain.isReplaying()) {
                driveTrain.stopReplay();
            }
            driveTrain.toggleRecording();
        }

        if (oi.getDriveYButton()) {
            if (driveTrain.isReplaying()) {
                driveTrain.stopReplay();
            } else if (!driveTrain.isRecording()) {
                driveTrain.startReplay();
            }
        }

        if (driveTrain.isReplaying()) {
            driveTrain.replayStep();
        } else {
            double forward = -oi.getLeftDriveY();
            double turn = oi.getLeftDriveX();

            driveTrain.driveArcade(turn, forward);
            driveTrain.samplePath();
        }
    }
}
```

PathRecorder.java: replayStep()

```java
public double[] replayStep(double currentLeft, double currentRight) {
        if (!replaying || replayIndex >= currentSamples().size()) {
            replaying = false;
            return null;
        }

        Sample s = currentSamples().get(replayIndex);

        boolean commanding = Math.abs(s.leftPower) > 0.05 ||
Math.abs(s.rightPower) > 0.05;
        if (replayIndex > 0 && commanding && checkStall(currentLeft,
currentRight)) {
            System.out.println("Stall detected on path " + (currentPathIndex +
1)
                    + " at sample " + replayIndex + ": stopping replay.");
            replaying = false;
            return null;
        }

        if (!commanding) {
            stallCounter = 0;
            stallPrevLeft  = currentLeft;
            stallPrevRight = currentRight;
        }

        replayIndex++;

        return new double[] { s.leftPower, s.rightPower };
    }
```

DriveTrain.java: replayStep()

```java
public boolean replayStep() {
    double[] powers = pathRecorder.replayStep(
            getLeftEncoderDistance(), getRightEncoderDistance());
    if (powers == null) {
        driveTank(0, 0);
        System.out.println("[PathRecorder] Replay finished on path " +
(pathRecorder.getCurrentPathIndex() + 1) + ".");
        return false;
    }
    driveTank(powers[0], powers[1]);
    return true;
}
```

DriveTrain.java: toggleRecording()

```java
public boolean toggleRecording() {
    if (pathRecorder.isRecording()) {
        pathRecorder.stopRecording();
        System.out.println("[PathRecorder] Stopped recording path " +
(pathRecorder.getCurrentPathIndex() + 1) + " - " +
pathRecorder.getSampleCount() + " samples saved.");
        return false;
    } else {
        pathRecorder.startRecording();
        System.out.println("[PathRecorder] Recording started on path " +
(pathRecorder.getCurrentPathIndex() + 1) + ".");
        return true;
    }
}
```