Team Sidekick
Project SIDEKICK
Final Design Report

05/16/2025

Executive Summary (EXSUM)

One of the main problems faced by planetary exploration today is the mobility limitations of current vehicles, specifically in regards to steep valleys and cliffs. These limitations have stopped space agencies from visiting certain craters and regions they deem too dangerous for their rovers, preventing scientists from getting a full picture of Mars' geological past. The space exploration community is in need of new, novel technologies to solve this problem.

Our team is proposing a new type of vehicle, one that could safely explore areas of Mars that were once deemed inaccessible. This would take the form of a small rover, and it would be directly tethered to a larger rover similar to those currently on Mars. This small rover, which we have called "Sidekick", would be lowered down a valley or crater using the tether and an electrical winch. Sidekick would be able to perform analysis on the cliff face as it descended, taking data and images of the geological features. Then, once the tether length was reached, the larger rover could use the winch to reel Sidekick back up to the surface.

With the funding and timeline our team was provided, we developed a basic prototype of Sidekick, integrating specific design features that could help the vehicle scale dramatic inclines safely and efficiently. Some of these features are mechanical, such as internal spring suspension for the rover's payload and instruments, while others are more software-based, like a set of motorized arm joints that are meant to automatically conform the wheel positions to the tilt of the terrain. Combining these two systems should result in a robust vehicle that can safely navigate and adapt to severe Martian terrain.

To test out this design on real terrain, we plan to travel to a nearby geographical feature that resembles a Martian crater or cliff. The prototyped Sidekick will be hooked up to a hand-powered winch and slowly lowered down a steep incline. We will record the prototype's motion and see if it can perform a controlled descent and ascent. We will also watch the rover's payload and see if our design innovations allowed it to maintain stability throughout the excursion. The observations taken during this test will inform any future design changes or iterations if we were to continue this project in the future.

The results of this test will be very significant, because if our design considerations seem to improve the rover's vertical mobility, then we could present Sidekick to NASA with the intention of developing it further. Our design would heavily address some of the mobility issues that NASA missions face, since many of their rovers have made it a point to avoid harsh terrain. Our design, if testing proves successful, could accompany one of NASA's future rovers. Sidekick could be sent to retrieve data down steep rock faces and crater walls without endangering the main rover, resulting in an exploratory mission that's safer and more scientifically lucrative.

Throughout our project's progression, we kept in mind both NASA and Texas Space Grant Consortium (TSGC), our two stakeholders. TSGC is our project sponsor, and provided us with the funds for the prototype so that we could compete in the TSGC Design Challenge. Therefore, our main priority was to have the prototype operational and presentable in time for the design challenge showcase. NASA, however, would be the primary consumer for Sidekick if it were to ever be fully developed. Thus, we made sure during the design phase to cater to NASA's design standards. If our project were to catch NASA's attention, we might gain their financial support or technical expertise, which would allow for the continuation of our project.

Table of Contents

Abstract

Mankind's ability to explore the surfaces of other planets has been severely limited by their dangerous terrain and the difficulty of sending exploratory vehicles from Earth. The current method to explore surfaces has been to send rovers to planets, such as Mars, to analyze and collect data. Recent missions such as the Perseverance rover have pushed this further by adding a companion vehicle, Ingenuity, to accompany the rover and explore more aspects of Mars' surface. Project Sidekick aims to further develop the idea of a companion vehicle by adding a smaller rover, or "Sidekick", to a larger main rover that can be sent to explore treacherous terrains. Sidekick is intended to be lowered down into steep craters on the surface of Mars by a winch and cable attached to the larger rover, sending images and data back up to the surface. To determine the viability of Sidekick, a small-scale prototype is being designed and constructed out of machined aluminum and off the shelf components. Once completed, Sidekick will be attached to a winch system modeling the large rover and tested on local geographical features that resemble those found on Mars.

1. Introduction and Overview

Modern humans have been on Earth for a couple hundred thousand years, yet our science and technology have only recently matured. Within the last 500 years, it was discovered that the Earth and other planets orbit our central star in a heliocentric manner. For less than 50, Mankind has used its titanic industrial might to carve a small piece of the most valuable and elusive resource ever conceived: perspective. Why are we here? Where did all of this come from and where is it going? While astronomers and physicists are focused on massive light-year scale star maps and invisible forces, geoscientists search for answers to these questions in a much more tangible way.

Geoscientists like our contact at UT's Jackson School, Dr. Timothy Goudge, draw conclusions from rock records preserved on planetary surfaces [1]. Dr. Goudge's research investigates the possibility that Ancient Mars harbored conditions capable of sustaining extraterrestrial life. The primary area of focus in his research is physical and chemical characterization of the available Martian rock record. These records are stored as rasterized images, topographic maps, thermal imagery, and mineral composition datasets. Unfortunately, the variety of information available for examination is limited by current data collection solutions.

Figure 1. ***Strata along the face of sedimentary rock formations.[2]***

Orbiters like NASA's Mars Reconnaissance Orbiter (MRO) [3] and the European Space Agency's (ESA) Mars Express Orbiter (MEX) house high-resolution cameras, spectrometers, and radar scanners that allow for macro-scale surface and subsurface imaging and compositional analysis. Landers are stationed at safe landing sites capable of capturing meteorological conditions, Martian soil (AKA regolith) data, and seismic activity. Finally, rovers[7] provide mobile and higher-fidelity components to many of the same instrumentation found on orbiters and landers. Rovers, however, cannot travel everywhere [4]. Despite their advanced dynamic systems and material compositions, they have been locked into traveling through mild terrain with few obstacles and shallow slopes. These travel restrictions prevent up-close analysis of strata (Figure 1) engrained along sedimentary rock boundaries that could be used to tell stories of Mars' past relationship with liquid water. NASA's Space Technology Mission Directorate has taken note of this issue and identified the problem as shortfall C3-1336: "*Robotic Mobility for Robust, Repeatable Access to/through Extreme Terrain, Surface Topography & Harsh Environmental Conditions"*[5]. This robotic mobility issue is one of dozens of shortfalls currently facing NASA's engineers. To accelerate technological innovation in robotics, material science, avionics, and much more, NASA hosts an open design competition dubbed "TechLeap."

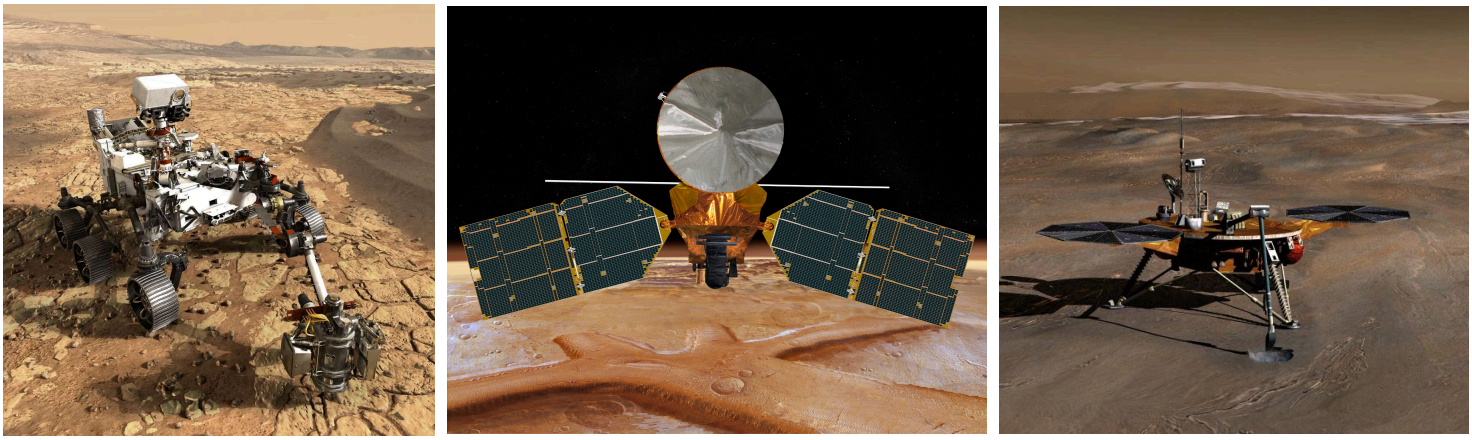


Figure 2. ***(From left to right) Perseverance Rover[6], MRO[3], and the Phoenix Lander[7]***

The NASA TechLeap competition [8] requires applicants to undergo a lengthy application process including technical writing, video presentation, and financial analysis for future development. If a panel of experts deem the showcased technology worth investment, a prize will be awarded to ensure further research and development. Our group has opened an application for NASA TechLeap thanks to the guidance of our sponsor, the Texas Space Grant Consortium (TSGC). TSGC is a UT-based organization with the objective of ensuring that the benefits of space research and technology are accessible to all Texans. TSGC has allocated funds for our group to develop a low-fidelity prototype of what a solution to our selected robotics shortfall might look like. Furthermore, TSGC mandates their own deliverables technical and nontechnical submission materials. To aid our technical design, our group has appointed Dr. Junmin Wang to serve as faculty advisor. Dr. Wang conducts research focused around dynamic modeling with further emphasis on vehicle systems and robotics. Dr. Wang and Dr. Goudge provided us access to documentation including topographic data for specific landing sites, simulink control system libraries/documentation, and a similar concept proposal published by NASA JPL a few years ago.

This project was recently unveiled during TSGC's Spring 2025 conference. Over the last few months, we created a competition profile with TSGC, located resources, formed advisor relationships, designed, and manufactured the initial prototype. This project was able to stay on a strict timeline using the Gantt Chart that we constructed early on. Moreover, we held biweekly technical advisor meetings, and sent out weekly updates to staff at TSGC. This accumulation of resources allowed us to deliver a solution that covered NASA's specified problem criteria and exceeded our expectations given such a short timeline.

2. Problem Statement

From the challenges posed by the NASA Techleap competition, our group selected: *Robotic Mobility for Robust, Repeatable Access to/through Extreme Terrain, Surface Topography & Harsh Environmental Conditions*. Rovers normally weigh close to a ton and have a large

surface area, making it difficult to traverse various environments such as craters, cliffs, and caves, which limits their area of investigation and data collection. Normally, rovers either avoid these areas altogether or take prolonged periods to maneuver toward their destination. Attempting to explore these regions can cause the rover to suffer critical damage, and in some cases, the mission can be lost altogether.

A few immediate goals we confronted are the deadlines posted by the Texas Space Grant Consortium (TSGC), our sponsor. The final prototype submission was due April 17th, along with other check-ups along the way, giving us a short window to choose our approach and develop a proper prototype. We were given $1,200 by TSGC to build our prototype, meaning that we want to keep our Bill of Materials within this budget. We aimed to develop a companion robot which helps host rovers traverse the more treacherous terrain on Mars, specifically, craters such as the Oyama Crater [9], Muara Crater, and Moreux Crater, which are all located in the Mawrth Vallis region of Mars. This robot is designed to be disposable, but also durable enough to survive harsh missions. This new rover design would be able to maneuver in areas where a traditional rover cannot. It must also be able to transmit data to the host rover and work independently so that both vehicles can perform missions simultaneously.

We planned to develop our companion robot, Sidekick, with a unique controls-centric driving system that allows the robot to change ride height while driving. The robot also would come equipped with a set of wheels capable of moving along rocky surfaces. For structural design, we have opted to use aluminum t-slot beams and other aluminum plates and tubing to fabricate the majority of our rover. For the payload, which is meant to carry our instrumentation, we have opted for a mix of laser-cut acrylic and wood. These materials were chosen largely for compliance, accessibility, rigidity, and in the case of the payload, price point. The structure of the vehicle is square, as we wanted to keep the geometry simple for ease of manufacturability, assembly, and maintenance for our first prototype. In our initial design, we wanted to incorporate a cable that attaches the companion to the larger rover for data and power transmission however, we did not have access to a rover or the resources to develop the cable at the time, so we opted to use a steel cable winch as a proof of concept. Our priority with this companion robot was to emphasize the unique drivetrain. We hoped to use our robot to crawl down crater walls and perform analyses on the rock face using instruments suggested by Dr. Goudge. Due to budgeting and time constraints, we believed those features will be implemented later. We planned to debut our prototype's abilities in a video taken at Enchanted Rock State Park, where we would prove our robot was capable of driving along steep rocky surfaces.

By the end of this project, our main goal was to have developed a serviceable prototype for the competition. We hope to elaborate on this prototype at a later time given a larger budget and time window from NASA. Our thought was that if our prototype was successful enough at maneuvering harsher terrains, the next step would be to include higher-spec instrumentation, use proper aerospace-grade materials, and develop a power-data transmitting cable that connects our companion to the host rover. Overall, the final goal was to prove our robot's ability to fit within the design challenge criteria.

3. Design Alternatives

3.1 Leading Design

Considering the design goals previously mentioned, Sidekick needed to have off-road capabilities with a four wheel drive system while minimizing weight without sacrificing robustness. As a result, our leading design was based around a 6061 aluminum structural components including T-Slot extrusions, rectangular tubing, and panels. Starting at the center and moving outward, the assembly shown in Figure 3 consists of a central payload box, structural frame, and drive arms. The rigidity of the system comes from the frame that surrounds the instrumentation onboard the payload while still providing stability and overall functionality. Attached to this external frame will be four arms that are our drive legs. Each drive leg can be actuated by one of two motors that is housed inside the frame to allow the body to increase and decrease its center of gravity by raising and lowering the frame and the payload body. At the end of each drive leg is a specialized wheel that maximizes rigidity with compatibility for rough terrestrial terrain with a leading material choice of thermoplastic polyurethane (TPU). The drive motor, like the arm actuation motor, is housed inside the frame of the arm that connects the wheel to the main chassis. Existing technologies that have shown success include the Mars Sojourner Rover [10]. The Mars Sojourner Rover's profile is a small box with 6 wheels closely attached to the body [10]. Sojourner weighs 23 pounds which puts us at an extremely similar weight and design methodology [10].

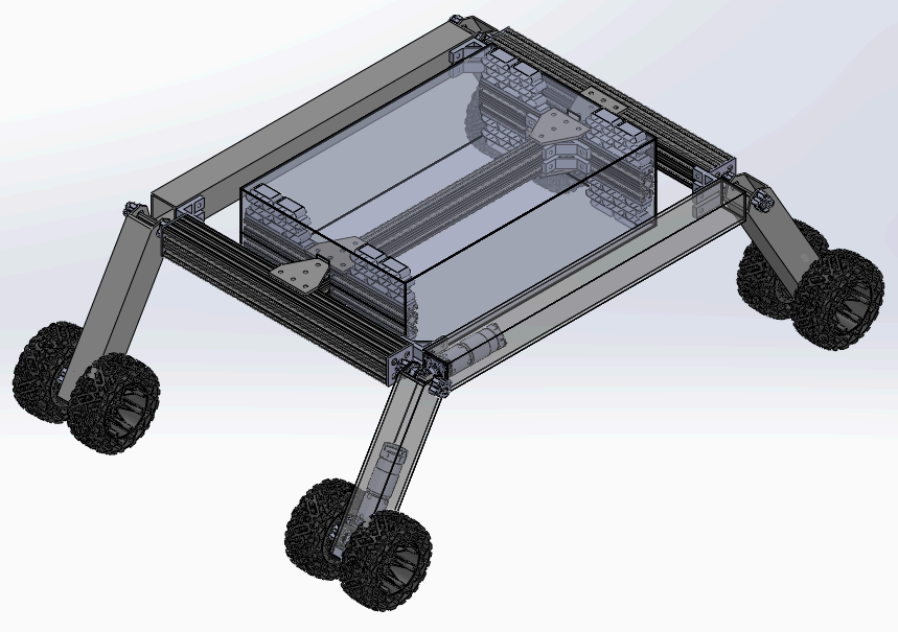


Figure 3. Near Finalized Sidekick Assembly

3.2 Frame and Chassis

The inner frame design follows the shape of the letter I enclosed by the outer frame (Appendix B). The entirety of the frame will be a combination of 6061 t-slot aluminum extrusions and hollow square extrusions. Appendix B demonstrates the design for shape and configuration of the frame. Each joint was connected using a variety of brackets and fittings that maximize strength and minimize stress, emphasizing the rigidity of all corners and mating

connections. Existing applications and inspiration stems from a turtle rover being funded on kickstarter that has a central payload surrounded by a frame to protect the vehicle [11].

3.3 Payload and Suspension

The structure of the payload is a simple box design (Appendix C), made of wooden panels and connected by 3D printed corner brackets. The top of this box is made of transparent acrylic so that the internals can be monitored at all times. The bottom of this box is where the electronics are mounted. Using accessible external brackets, these trays are easily detachable. This simplifies assembly and allows for quick access to the electrical system if it needs to be updated or repaired.

Considering the vehicle will be exploring treacherous terrain, we implemented an internalized form of suspension that can keep the payload stable throughout the journey. This will ensure that the rover's instrumentation remains intact as we go over bumps and expose the vehicle to sudden motion shock. The suspension consists of four custom printed springs (Appendix D) placed along the extrusions that shape the top and bottom forms of the letter I. The payload walls enclose these suspension springs to where they press up against the top and bottom plates of the payload. As the rover goes over rocky terrain, the springs will dampen the shaking of the payload, thus minimizing the vibrational force experienced by the onboard instruments as Sidekick descends an incline.

3.4 Drive Motor and Arm Actuation

Sidekick has 4 actuating arms, each with wheels attached to help carefully traverse tough terrain. Just like the Perseverance rover, Sidekick has a motor to power each wheel [12], however, Sidekick's motors differ in their placement. Instead of being directly attached to the wheel, instead, the motors are placed along the length of the arm and steel bevel gears are used to transfer the motion perpendicularly to the wheels. This setup is demonstrated in Appendix C. This design allows Sidekick to have a more streamlined figure to help it descend into craters and take minimal damage. These motors are housed in 6061 aluminum square tubes that attach to the frame/chassis (Fig. 3).

To complete the design of the arms, bearings are placed in the aluminum tubing to allow for the wheels to roll smoothly and aluminum hubs are screwed onto the axle to directly attach the 3D printed wheels to the arms. The top of the arm has two aluminum hubs directly screwed to it to allow for it to easily slide onto the chassis during construction. Two 45 degree angle cuts were made at this connection point on the aluminum tubing to ensure there is no interference with the chassis while the arm is moving. A completed arm assembly can be seen in Figure 4.

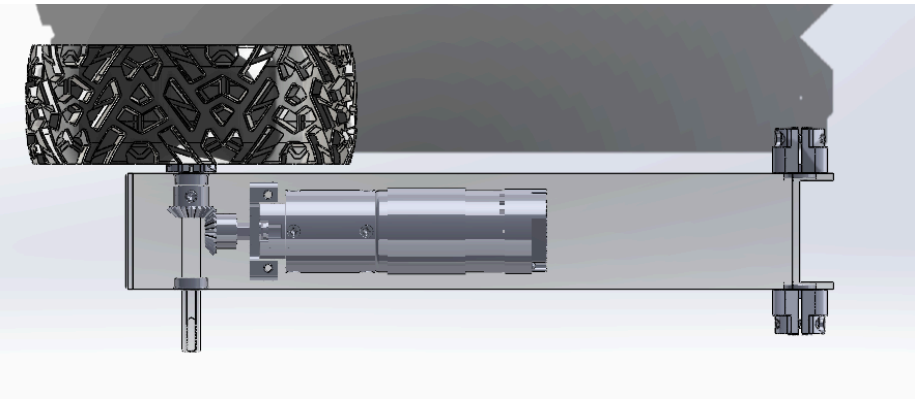


Figure 4. Completed Arm Assembly

Unlike Perseverance, Sidekick does not have dedicated steering motors [12], but does have an additional form of actuation to aid in maneuvering through terrain. As previously mentioned, two motors are used to actuate the arms, one for the back and one for the front arms. This allows Sidekick to raise and lower its arms to change its center of gravity which helps keep it balanced and drive over rugged terrains. The motors are housed in the 6061 aluminum square tubes in the chassis with only one motor being responsible for two arms to help reduce the costs of the prototype and make implementing a control system easier. For a final product, Sidekick would ideally have a motor for each arm allowing them to each actuate independently. The same bevel gear system is being used to transfer motion to the arms with a long axle running through the frame to transfer motion to both arms on either side. The motors used for both the wheels and the arms will be relatively high in torque (250 kg cm) allowing for better control and precise movements which is essential for Sidekick as it slowly descends into craters.

All four aluminum square tubes were machined in the UT ME machine shop. Holes have been added to allow for axles and motors to be mounted in exact locations allowing the four arms to be identical (Appendix F). Each arm was assembled by attaching a motor mount to each motor and a bevel gear. This sub assembly was then placed inside the aluminum tube which has two bearings press fit onto two sides to allow for the axle to be placed with the perpendicular bevel gear. The 3D printed wheels were then attached to the axle using a shaft collar. The rims were printed out of PETG and the tires out of TPU to allow for compliance. The top of the arms then had additional shaft collars added to the top to allow for easy integration to the frame axle. The completed arm assembly can be seen in Appendix G.

For the future, while a control system is planned on being implemented, it may not be enough to mitigate the effects of the crater's terrain on Sidekick. If testing finds that Sidekick is unable to traverse terrains smoothly enough, spring-dampers will be incorporated into the arms to aid the suspension system in the payload. By adding additional suspension to the arms, the amount of vibrations that Sidekick will be subjected to will decrease allowing increasing the longevity of the rover. The spring-dampers can easily be incorporated later on in the design process as needed by cutting the arms in half and adding inserts on both pieces so the spring-damper can be used to attach them back together. A design has not yet been developed due to the uncertainty of its necessity. Another area that needs to be explored is the connection between the arms and the frame. While the physical and motion connection are completed, the bevel gear system is exposed (Fig. 3) to the elements which in a Martian environment, could quickly lead to system

failure due to dust getting into the gears and interfering with their motion [13]. This could potentially be done by adding a strong yet flexible material to bridge the gap, however, more research is needed to best determine the solution to this issue. Additionally, creating larger wheels could help traverse terrain better as well and due to the amount of time it took to print, we were only able to print 4 out of the 8 wheels originally planned. Having 2 wheels per arm would aid in supporting the weight of Sidekick and further ensuring the arms are strong enough to hold the frame and payload.

3.5 Electrical Design

Sidekick's complex electrical system was necessary to enable robust power delivery, reliable remote operation, and rapid-response closed-loop control regimes. These primary functions dictated five categories of components amongst Sidekick's bill of materials. Sidekick's onboard electronics were a low-cost representation of real aerospace-grade electronics. In particular, this prototype took inspiration from open-source quadruped robot configurations[14]. Foremost, a source of power was absolutely necessary for running any electronics. Since the robot was designed to be anchored via a winch system, power could be distributed through an extension cord that sat alongside the existing steel cable. The full power distribution subsystem included a solar generator (temporarily acquired for field testing), the extension cord, and a 12V 30A AC-DC power supply unit (PSU). The PSU fed to two separate terminals that behaved as 12V and 5V lines. Succeeding the 12V terminal was the next critical component: an arduino microcontroller. Logical input served an essential role in allowing feedback for the closed-loop control system [15]. With 5V and 12V input, Sidekick's ESP-32S could manage user input/output signals, while arduino mega could receive sensor input and send motor output commands [16] and make automatic adjustments determined by algorithms flashed onto its memory. The microcontrollers received input signals from an array of analog sensors. 6 encoders distributed amongst all motors returned positioning information, an ultrasonic rangefinder housed beneath Sidekick's underbelly provided ground distance information[17], and an onboard accelerometer shared orientation, acceleration, and velocity metrics. Using this accumulated data, the arduino mega sent logic signals to 3 dual-motor controllers which in-turn supplied brushed motors the full 12V DC signal from the PSU. Finally, the user was able to receive physical controller feedback signals that communicated safety warnings through rumble, color, and text. Together, the power supply, logic controllers, analog sensors, brushed motors, and bluetooth feedback system produced an array of electronics (Fig. 5) that permitted full utilization of Sidekick's mechanical agility.

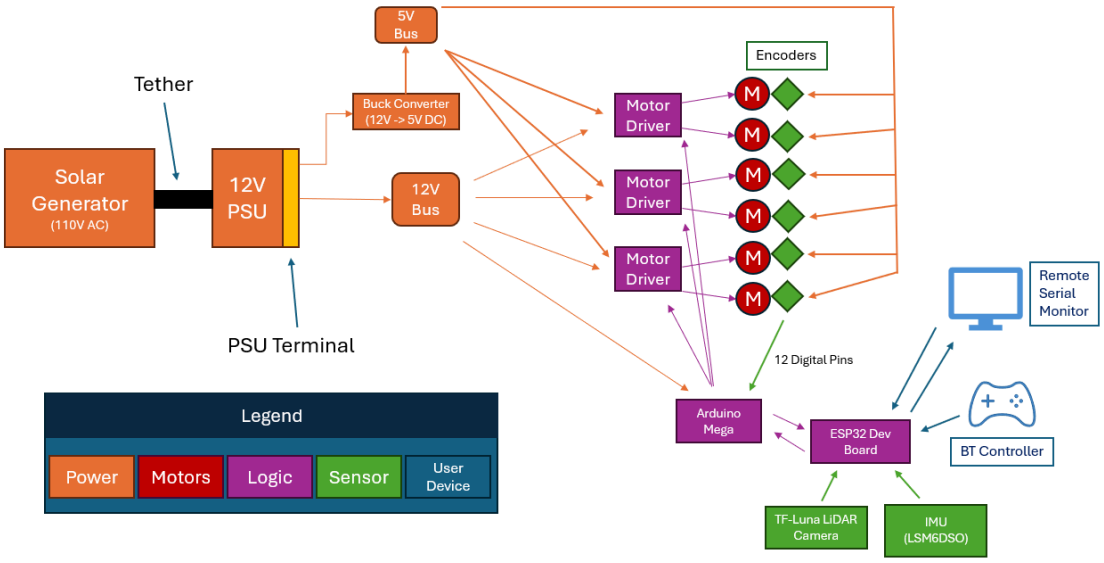


Figure 5. Sidekick Electrical System Flowchart

3.6 Controls

The unforgiving topography found while scaling steep terrain necessitated Sidekick to: (a) be aware of the main body's ground clearance, (b) make slow and precise movements, and (c) actively stabilize to reduce mechanical strain on the delicate electronics. It was determined from an early stage that these requirements were indicative of a closed-loop control system. However, an issue arose in that closed-loop control systems demand specific reference vectors while the user would only be equipped with short serial commands or a handheld controller. Therefore, Sidekick utilized a complex C++ script (Appendix) flashed onboard its microcontrollers. Sidekick's firmware could be interpreted as having three distinct layers: interpretation, calculation, and execution. Arduino Mega (Appendix) used an array of digital and analog ports, but they each needed to be configured to send or to receive 5V logic signals. Moreover, various components required different signal types. The DC motor drivers required PWM-capable pins, encoders demanded interrupt-capable pins, and receivers desired 3.3V rather than the nominal 5V. Sidekick's first layer allowed the arduino to make all of these connections correctly and accurately, and its code chunks were composed of an array of pinMode functions. Subsequently, the microcontrollers could read and print incoming data to all pins specified as inputs. In the second firmware layer, these datasets were interpreted and transformed into output signals. Data including ground distance, motor position, and body orientation vectors were subtracted from set reference values to extrapolate error. For ground distance, a high/med/low drive height could be toggled via remote control. Bumps, rocks, and dips would result in a sudden change of the real ultrasonic distance value against the reference ride height, resulting in a rise in the error value. Similarly, real motor position and velocities were tied to thumbstick controls, and inaccuracies

recorded as errors by the motor encoders. Body pitch reference values were adjusted for hill ascent/descent using a wheel input (also on external controller). All error information was transformed into the output signal through a proportional-integral-derivative (PID) function. By adjusting coefficients in the true error, derivative error, and integral error components of these three datasets, all three possible deviations from expected drive behavior could be corrected for in the final driving scheme. In the last layer of firmware, Sidekick took the array of signal functions from this calculation and executed them as voltage output signals for the digital and PWM pins. This signal was read by the motor controllers and adjusted motor behavior accordingly. Ultimately, the closed-loop regime Sidekick employed allowed for safer and easier operation.

4. Design Evaluation and Analysis

4.1 Criteria

Collectively we decided on a set of criteria that would deem our rover Sidekick as a successful product. The first being a compact form factor. Considering Sidekick is meant to be a companion rover that deploys from a primary larger rover, it is vital in keeping the rover as lightweight and compact as possible. More specifically, we wanted the rover to be under 50 lbs and within a size of 2.5 ft long by 2 ft wide. As a result, the frame design is meant to maximize those proclaimed limitations with space for a compact and robust center payload with all the instrumentation. To analyze stresses on the frame, we conducted finite element analysis at different points of the rover to ensure stability at nominal gravitation forces and sudden contact with potential rocks or walls.

Another success criteria involved Sidekick being as low-cost as possible considering it is meant to be deployable and disposable in the event of failure. Sidekick should pose no risk to the primary mission rover and keep it low-cost which minimizes risk and increases overall longevity of funded missions like ourselves. Our goal was to keep our Sidekick prototype under $1200. At $1200 we had flexibility in material choice, motor specifications, and overall robustness by the availability of consumer off the shelf products at this price point. To stay within this budget, a majority of the rover was made of 6061 aluminum with a combination of 3D printed components where necessary. The largest portions of the rover are the frame, payload, and arms, all of which are 6061 aluminum to maximize robustness while minimizing cost.

4.2 Performance

Performance is an extremely delicate and important aspect of Sidekick. On top of being a general drive rover, Sidekick is able to change its center of gravity by controlling the height at which the payload sits. This is accomplished through arm actuation. In simple terms, the length of the arms is fixed but the angle at which they are relative to the payload can range from a 0 degree bench mark up to 180 degrees. This range of motion allows for the implementation of control systems with a gyroscope as another form of stabilization and suspension as we traverse rigorous terrain. Because the arms have 180 degree range of motion, they can easily and accurately scale rocks, hills, and sudden changes in incline with ease. Being able to lower the

payload's center of gravity depending on the environment it is in allows for more gradual and smooth descent as it scales cliffs and craters.

4.3 Feasibility

Feasibility is something that we prioritized and considered heavily as we made design decisions. As previously mentioned, a majority of the rover is made of consumer off the shelf products with some components having been designed and manufactured in house. Because the parts were mostly aluminum, any modifications we made were easily done in the Mechanical Engineering machine shop. The most bold approach in achieving performance was the development of the suspension. There is one leading form of suspension on Sidekick with future plans to iterate another using shock absorbers. The primary form is inside the payload box, suspending the payload from the frame on both top and bottom faces of the aluminum extrusions. This ensures the payload does not experience shock and can distribute the sudden bump or shock forces. To increase reliability, we are also planning to include spring shock absorbers as a future iteration of the arms to distribute shock forces as we run over rocky terrain on Mars. Apart from these two mechanical suspension systems, we also implemented control responses to actuate the arms based on feedback from a gyrometer embedded in the arms. This allows for each set of arms to be actuated automatically based on obstacles that need to be traversed to ensure the payload remains at a level height while allowing the chassis to respond according to the environment. The largest risk in this approach was the time constraint to implement such a feature. However, because it is an electronics issue and would not directly affect the functionality of the rover as we can implement it on top of the mechanical system. Through several challenges, we were able to successfully build all major design goals and components into our prototype. By staying within these feasibility constraints, we were able to successfully create a remotely operating repelling rover with drive capability.

5. Testing Approach

5.1 Static Body Test - Frame & Payload

Static body tests on Sidekick’s frame and payload are intended to verify the structural integrity and load-bearing capability of the rover chassis under static load conditions representative of normal terrain operations. To make sure Sidekick’s frame can support the expected weight of the payload and survive forces it may feel during launch, landing, or while climbing over rocks, we performed several different tests at different conditions to simulate a normal operational environment. To replicate environmental stresses, we conducted field tests in rocky terrains and steep slopes around the UT campus.. These physical tests confirmed structural integrity and identified areas for design improvement that could be implemented in the future.

To test the frame’s structural capabilities, we simulated a larger than expected payload mass of 20 pounds to validate the load capacity. Applying artificial weights to the payload gave us the maximum payload weight we could apply while still achieving complete functionality. After each load test configuration, we checked the structural response using our eyes with brief ruler measurements to see how much the part deflects before and after the load is applied. The maximum deflection we observed was approximately 1.5 inches which was not enough for us to have significant doubt in our payload design. Photos were taken from the side along with a

timelapse to more accurately analyze deflection. Mounting points including bolts and joints were also inspected to see if anything loosened or shifted.

To test the frames integrity while under high vibrational forces, such as those experienced during certain stages of launch, our testing procedure had to be creative. We do not have access to the vibration testing rigs commonly used in the space industry, so we will have to use the resources available to us to generate a relatively comparable level of vibration. In our case, we vibrated the prototype manually, shaking it in the X,Y, and Z directions to see if any components deform or loosen. We also had the prototype undergo the vibrational forces of driving on a rocky asphalt road. Although Sidekick is unlikely to ever drive across this type of terrain, this generated a significant amount of vibration that put our hardware to the test. Although this was the extent of our vibrational testing, we hope to eventually use the accelerometer onboard the rover to quantify the vibrational forces created by both strategies, and compare them to official NASA testing standards which can be found publically. If the magnitudes are close to matching, we can be confident that the assembly would be able to withstand the vibrations of launch.

To test how the rover frame handles twisting and uneven pressure from rocky terrain that we are expecting to encounter or Mars we tested in similar real-world situations. We used the UT campus to test our structural integrity. We added the same weight used in the static test to simulate larger than expected payload mass and secure it to prevent shifting during movement. We then remotely drove the rover across the terrain and observed subtle flex and movement. We determined the frame handles shifting loads efficiently – paying close attention to if it creaks, flexes, or seems unstable. We also recorded slow-motion side-angle views to determine any chassis flexes when going over big rocks or dips. We particularly looked at connecting and mounting points to see if twisting caused any loosening or stress marks. The results from the tests just mentioned proved that our system was strong, reliable, and consistent. There were no abnormalities and all physical systems performed as necessary in the drive tests. Something we plan to do in the future but did not get to test on this prototype was simulating a landing procedure. To do so we would raise one wheel approximately 6-12 inches off the ground (on a brick or block), and then quickly remove the block and let it drop. This mimics a hard landing or rock slip, adding extra shock to a single corner of the frame. Similarly, we will observe and analyze how the rover handles the sudden forces and inspect the assembly again.

To conclude, our ideologies for these tests note that for all tests, safety is our priority. Wearing gloves, avoiding pinching hands under weight, and working in pairs was a necessity to testing the vehicle. In these tests we operated with heavier than expected loads to build confidence in our design. Repeating tests after modifications to ensure no new additions introduced weaknesses and maintaining written records to keep track of changes will help us succeed in frame and payload testing.

5.2 Drive Motor and Arm Actuation

With the prototype completed, we needed to analyze the performance of our motors in the arm actuation and driving functions of the robot. By defining a threshold of angles the arms are expected to be able to reach, we designed a set of experiments in which the robot is solely expected to move the arms up/down and hold the position for a specific amount of time.

The initial tests are to test a wide range of angle combinations for the arm actuation. The arms are designed to be able to actuate between 0 and 90 degrees and must be able to function

anywhere between this range on both arms. To test this, each arm is tested in increments of 10 degrees starting at 5 degrees up till 85 degrees. Combinations of each arm are tested in order to test each possibility for the arm positions. Some examples include each arm tested at the same angle (Ex. 35 degrees for both arms) and different angles for either set of arms (25 degrees for front arms, 75 degrees for back arms). In order to pass this initial test, each orientation needs to remain stable and be able to drive with the drive motors. These tests have not yet been conducted due to some stability issues with the bevel gear system for the arms. Once this issue has been addressed, these tests can be conducted in future work.

The drive motors and their wheels are tested through the previous tests on arm angle actuation to determine their functionality, but additional tests are performed to further evaluate their capabilities. The rover must have the ability to traverse different types of terrains and has been tested to do so. Initial tests have been performed where the rover successfully drove across tile floors, carpeted floors, and gravel sidewalks. Sidekick was able to go forward and backwards on these terrains without experiencing any issues. Future terrains that Sidekick will be tested on include grass areas, hilly areas, and areas with rocks, craters, and uneven terrain to demonstrate its wide range of capabilities.

Finally, it will be important to test the rover's capability to drive up and down different angled inclines. The planned tests start at 10 degrees and go up to 70 degrees in increments of 10 degrees. As long as the rover is able to descend and ascend the cliff, it passes the test. The arms of the rover are positioned to however is most optimal to do so and the speed of the rover is adjusted to allow for safe travel of the rover. By completing these tests, it will prove the reliability of the arms and drive motors and show they are robust and durable. While the plan for these experiments has been determined, they will be part of the future work since as mentioned earlier, more work needs to be done to improve the arm actuation system. The final testing site for Sidekick to perform all of its mobility functions in an environment similar to Mars is planned to be Enchanted Rock park due to its many steep inclines and variety in smooth and rough terrains that would allow us to demonstrate Sidekick's robustness.

5.3 Control Logic

We finally planned to test the controlling logic of our system. This included: gyroscope and ultrasonic range finder feedback, steering, propulsion, braking, and toggling the LED light. Most of these experiments were based solely on our own estimation of what would be acceptable. Unfortunately, due to approaching deadlines, being able to drive the vehicle in a longitudinal path was deemed acceptable for presentation. With that being said, the following protocols are what we hope to accomplish in the near future.

The feedback is only present within the closed-loop system, so by attempting to place our rover in situations, we expect the feedback would prompt certain motion. For example, our range finders are meant to ensure that the bottom of the rover body would never collide with the ground by causing the arm actuators to lift the body up. To cause this, we will drive over certain ridges or try to manually set the position of the robot, so that the range finders would prompt the arms to activate. The gyroscope performs a similar function, meant to help prevent the rover from flipping over, so we would then try to drive over steep rock faces to see if the arms react appropriately.

Steering, propulsion, and braking are coupled together by a Playstation 5 controller. Using the Playstation controller, we will be able to prompt the drive motors to push the rover forward. A braking function has also been implemented which causes the motors to stall. After confirming the desired function of the drive motors, we set out to elaborate on the testing of our drive train. To further analyze propulsion capabilities, we can set out a straight path and time the rover takes to travel from beginning to end. We can then estimate acceleration and velocity of the body. In doing so, we may also test braking distance and efficacy of the vehicle when at full speed, whatever that velocity value may be. Since the rover does not use a split axle, we do not have access to traditional motor vehicle steering. To turn left, the left motors are stalled while the right motors are activated, and to turn right, the opposite occurs. We would then test our steering by evaluating the rover's ability to pivot around obstacles as well as analyze the radius at which the vehicle can turn. Finally, our most simple test would be to test the toggling of our LED light bar. The plan is to include some code which controls sending a signal to the light to turn on or off. We would then assign a button on the controller to this function, then by simply pressing the button we would then be able to see if it was effective or not.

Moving forward, we propose a number of empirical metrics to quantitatively characterize Sidekick's performance:

Metric	Measurement Method	Goal	Unit
Ride Height Error	Range finder's measured deviation from setpoint	0	cm
Body Pitch Error	IMU's measured pitch angle deviation	0	°
Latency	Software measurement of signal input to output	250	ms
Traversal Success Rate	% successful runs without fault	80	%
Second Order Response Efficacy	1-10 qualitative rating of desired overshoot, steady-state error, and settling time. Data collected and analyzed after a powered run.	8	N/A

Table 1. Controls DOE Metrics

Equipment Tests	
1. Ride Height Regulation	
Required Sensors	Ultrasonic Rangefinder
Setup	Place Sidekick on (1) tile, (2) rock. And (3) gravel. Additionally Sidekick should be set to (1) high and (2) low ride height on each surface.
Procedure	1. Measure initial ride height

	2. Drive Sidekick over a bump or dip on the same surface 3. Measure final ride height
Output Metric	Ride height error (cm) will illustrate how well the system maintains its set safety distance from the ground. Ideally, the two ride heights would be identical across all surfaces (error = 0).
2. Stability Response Test	
Setup	Place Sidekick on a (1) flat surface, (2) shallow incline [5-10°], and (3) steep incline [30-45°].
Procedure	1. Measure initial pitch from IMU 2. Drive Sidekick over a bump or dip on the same surface 3. Measure final pitch from IMU
Output Metric	Body pitch error will illustrate how well the system maintains its set angle to prevent electronics jostling. Ideally, the two pitch offsets would be identical across all surfaces regardless of disturbance (error = 0).
3. Controller Latency Test	
Setup	Connect an external wifi-capable device (laptop, phone, etc.) to Sidekick's onboard host to remotely read system outputs. Ensure that the host is configured to output serial responses for desired action measurements.
Procedure	1. Drive forward for 3 seconds 2. Read and take note of recorded latency response on mobile device 3. Adjust ride height positions 4. Record latency response 5. Turn on headlight 6. Record response 7. Use personal device to send a message to Sidekick 8. Record latency reading for return response
Output Metric	System latency is directly proportional to electronic efficiency. Future optimizations are outlined by user input to response delays. An excellent response time is <250 ms.

Table 2. Controls DOE Tests

6. Outreach

At the end of this semester, we had the opportunity to present to the Walker Department of Mechanical Engineering's External Advisory Committee (EAC). The EAC is a group of UT alumni who are industry leaders and help provide the university with educational and industry guidance. Our professor selected our team to present to the EAC since one of the topics being discussed was about how to improve the senior design class for all future mechanical engineering

students. He was very pleased with the progress and scope we had obtained in our project over the course of the semester and wanted us to give our insights on the class, sharing what we enjoyed and what we would like to see improved in terms of the student experience. We were then presented with the opportunity to speak on behalf of our student body and improve the very system that brought us to this project. This was accomplished with a 10 minute presentation where we discussed the scope of our project and what we had achieved in the semester-long capstone course. Then we delineated the things we enjoyed about the class such as the standardized deliverables, which matched up well with our sponsor's deliverables and allowed us to adapt between them with ease. Next, we went over what we would change for future years, with a big part being that the capstone class might work better as a two semester course instead of just one.

Already, the results of our feedback have aligned and resonated with the mechanical engineering department's goals, and the capstone course will be continuing as a full-year class. We are hopeful that our input to the EAC will allow for the class to make further degree improvements that will serve to benefit mechanical engineering students in the future such that they can fully experience the design process before stepping into industry or academia.

7. Summary

Current rovers are limited by their ability to descend steep grades safely. As a result, valleys and craters have always been strictly off limits for martian exploration unless it was where the vehicle originally landed. To solve this problem, our team developed a new type of planetary exploration vehicle called "Sidekick" which could be deployed into these dangerous areas without risking the longevity of the mission. Connected by a powered tether to a main rover, Sidekick would be lowered into a crater or canyon to perform science and then pulled back out once the cable is exhausted. This vehicle has specific design considerations that allow it to more easily scale rock faces, such as automatically adjusting wheel heights and a spring suspension payload. To test Sidekick's viability, we created a low-fidelity prototype and tried to bring it to a rocky location like Enchanted Rock State Park. There, we would use a hand crank to observe the rover's vertical mobility along a rock wall. We then showcased these design innovations at TSGC's design competition, in the hopes that NASA might be interested in Sidekick and fund its development further.

8. Acknowledgments

We would like to acknowledge Dr. Junmin Wang Ph.D., our faculty advisor, for meeting with us regularly to weigh in on our project's progress. We greatly appreciate his recommendations regarding our control systems and mobility issues.

Additionally, we would like to acknowledge Dr. Timothy Goudge Ph.D. for providing his insight into the geological features of Mars and the ways that we can make our rover design distinct from those already in operation. Our discussion with him was very beneficial and helped us hone our project scope.

We would also like to acknowledge Dr. Timothy Urban Ph.D., our project sponsor, for his instruction and guidance regarding the TSGC and NASA design competitions. His flexibility and receptivity have been invaluable to our project's progress.

Finally, we would like to acknowledge the professors for our project course, Dr. Christopher Rylander and Dr. Joshua Keena, as well as our team's TA, Ali Ghasemkhani. We thank them for facilitating our project and assisting in its progression throughout the semester.

9. References

[1] "Assessing the Mineralogy of the Watershed and Fan Deposits of the Jezero Crater Paleolake System, Mars," Journal of Geophysical Research: Planets, vol. 120, no. 4, pp. 775-808, https://agupubs.onlinelibrary.wiley.com/doi/pdf/10.1002/2014JE004782 (accessed Feb. 17, 2025).

[2] "Geological strata: They're everywhere.," Creation Ministries International., https://creation.com/geological-strata (accessed Feb. 17, 2025).

[3] "Mars Reconnaissance Orbiter (MRO)," U.S. Geological Survey. (n.d.), Www.usgs.gov. https://www.usgs.gov/media/images/mars-reconnaissance-orbiter-mro-0 (accessed Feb. 17, 2025).

[4] "Premature Wear of the MSL Wheels," NASA JPL., llis.nasa.gov/lesson/22401 (accessed Feb. 17, 2025).

[5] "NASA Technology Strategy," NASA TechPort, https://techport.nasa.gov/strategy (accessed Feb. 17, 2025).

[6] "NASA's Perseverance rover has produced pure oxygen on Mars," MIT Technology Review,https://www.technologyreview.com/2021/04/22/1023404/mars-perseverance-rover-moxie-oxygen/ (accessed Feb. 17, 2025)

[7] "Mars Phoenix," Nasa. (n.d.), https://science.nasa.gov/mission/mars-phoenix/ (accessed Feb. 17, 2025)

[8] "NASA TechLeap Prize," Nasatechleap.org, https://www.nasatechleap.org/ (accessed Feb. 17, 2025)

[9] "Oyama Crater (ESP_038481_2035)," HiRISE, https://www.uahirise.org/ESP_038481_2035 (accessed Feb. 17, 2025)

[10] NASA, "Mars Pathfinder - NASA Science," *science.nasa.gov*, 2024. https://science.nasa.gov/mission/mars-pathfinder/(accessed Feb. 17, 2025)

[11] D. Etherington, "You can get your own mini Mars rover for Earth through this new project | TechCrunch," *TechCrunch*, Sep. 06, 2017.

https://techcrunch.com/2017/09/06/you-can-get-your-own-mini-mars-rover-for-earth-through-this-new-project/ (accessed Mar. 04, 2025).

[12] “Perseverance rover - wheels and legs,” LabXchange, https://www.labxchange.org/library/items/lb:LabXchange:f1c148d9:html:1#:~:text=The%20Mars%202020%20Perseverance%20Rover,place%2C%20a%20full%20360%20degrees. (accessed Mar. 3, 2025).

[13] “Scientists developing ways to mitigate dust problem for explorers,” NASA, https://www.nasa.gov/centers-and-facilities/kennedy/scientists-developing-ways-to-mitigate-dust-problem-for-explorers/ (accessed Mar. 3, 2025).

[14] “Dingo Quadruped,” Stanford Robotics Club, https://github.com/Yerbert/DingoQuadruped (accessed Mar. 3, 2025)

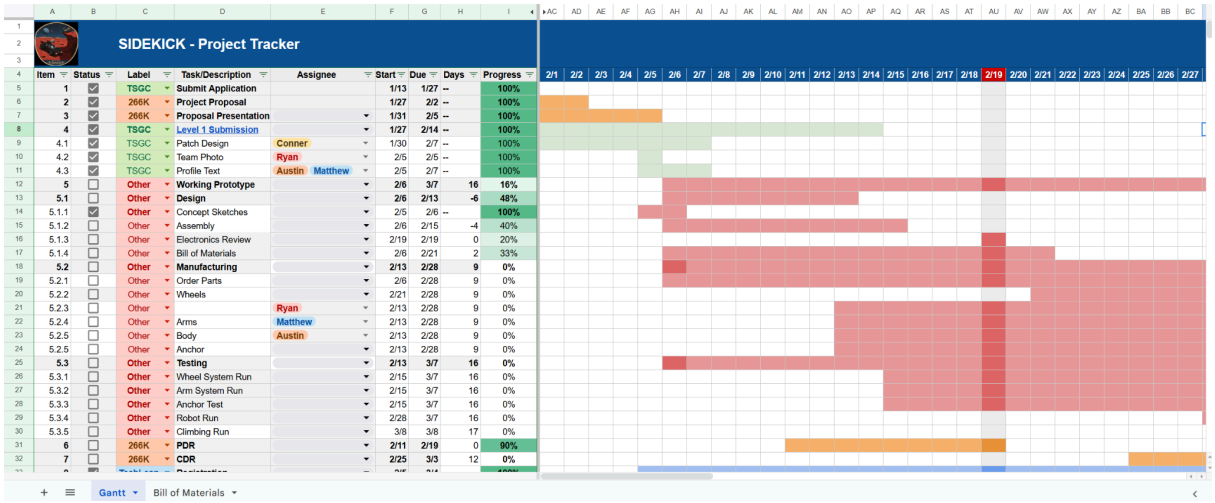
[15] “The PID Controller & Theory Explained,” National Instruments, https://www.ni.com/en/shop/labview/pid-theory-explained.html?srsltid=AfmBOop-HrUxCT5sUy40nlZ8cx5ML0tGuZoidKuOmrR1uegChE5NQ7Sp (accessed Mar. 3, 2025)

[16] “Basics of Pulse Width Modulation,” Arduino, LLC., https://docs.arduino.cc/learn/microcontrollers/analog-output/ (accessed Mar. 3, 2025)

[17] “The Basics of LiDAR - Light Detection and Ranging - Remote Sensing,” Neonscience, https://www.neonscience.org/resources/learning-hub/tutorials/lidar-basics (accessed Mar. 3, 2025)

Appendix

A. Gantt Chart



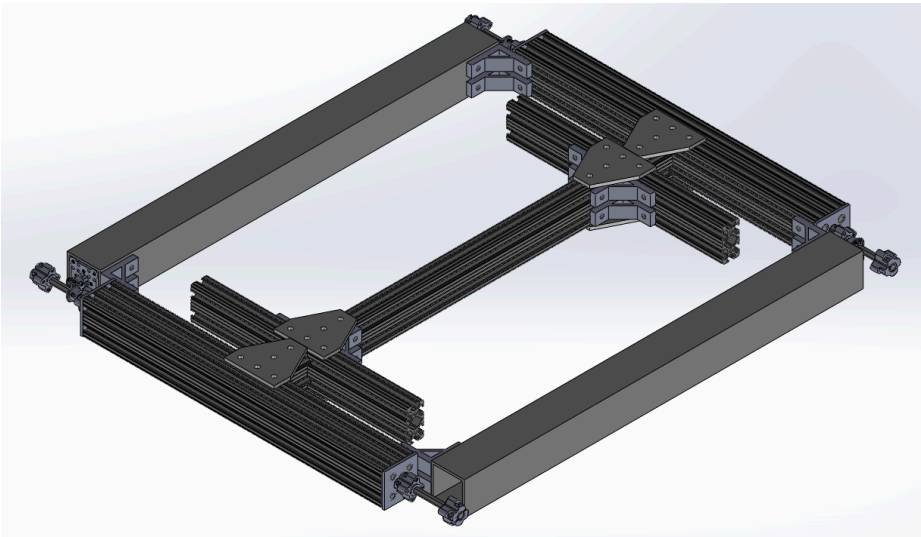
SIDEKICK - Project Tracker

Item	Status	Label	Task/Description	Assignee	Start	Due	Days	Progress
1	☑	TSGC	Submit Application		1/13	1/27	--	100%
2	☑	266K	Project Proposal		1/27	2/2	--	100%
3	☑	266K	Proposal Presentation		1/31	2/5	--	100%
4	☑	TSGC	Level 1 Submission		1/27	2/14	--	100%
4.1	☑	TSGC	Patch Design	Conner	1/30	2/7	--	100%
4.2	☑	TSGC	Team Photo	Ryan	2/5	2/5	--	100%
4.3	☑	TSGC	Profile Text	Austin Matthew	2/5	2/7	--	100%
5	☐	Other	Working Prototype		2/6	3/7	16	16%
5.1	☐	Other	Design		2/6	2/13	-6	48%
5.1.1	☑	Other	Concept Sketches		2/5	2/6	--	100%
5.1.2	☐	Other	Assembly		2/6	2/15	-4	40%
5.1.3	☐	Other	Electronics Review		2/19	2/19	0	20%
5.1.4	☐	Other	Bill of Materials		2/6	2/21	2	33%
5.2	☐	Other	Manufacturing		2/13	2/28	9	0%
5.2.1	☐	Other	Order Parts		2/6	2/28	9	0%
5.2.2	☐	Other	Wheels		2/21	2/28	9	0%
5.2.3	☐	Other		Ryan	2/13	2/28	9	0%
5.2.4	☐	Other	Arms	Matthew	2/13	2/28	9	0%
5.2.5	☐	Other	Body	Austin	2/13	2/28	9	0%
5.2.5	☐	Other	Anchor		2/13	2/28	9	0%
5.3	☐	Other	Testing		2/13	3/7	16	0%
5.3.1	☐	Other	Wheel System Run		2/15	3/7	16	0%
5.3.2	☐	Other	Arm System Run		2/15	3/7	16	0%
5.3.3	☐	Other	Anchor Test		2/15	3/7	16	0%
5.3.4	☐	Other	Robot Run		2/28	3/7	16	0%
5.3.5	☐	Other	Climbing Run		3/8	3/8	17	0%
6	☐	266K	PDR		2/11	2/19	0	90%
7	☐	266K	CDR		2/25	3/3	12	0%

Gantt Bill of Materials

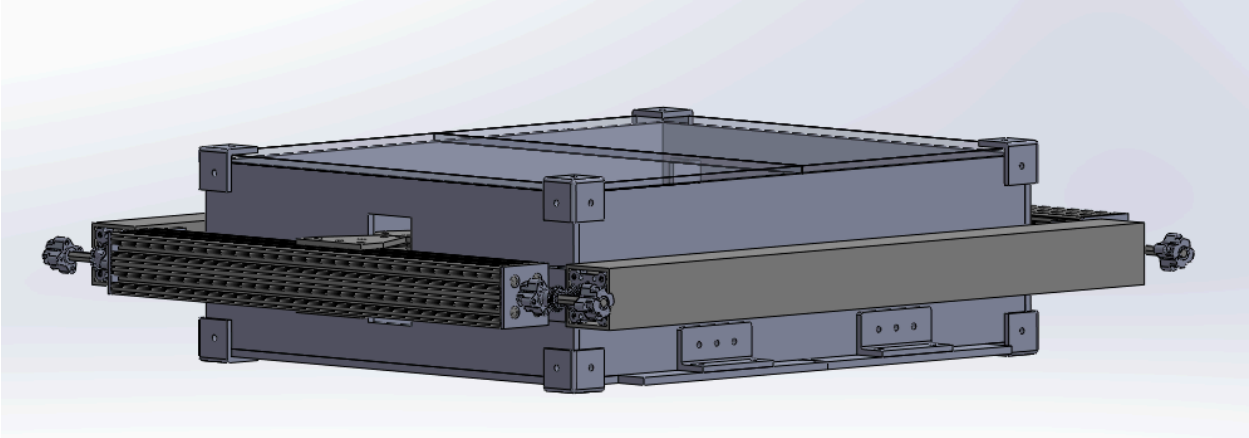
Gantt Chart detailing project schedule, competition deadlines, and other important dates regarding the project.

B. Chassis and Frame Finalized Design



The outer frame takes the shape of a rectangle that is connected to the inner chassis using a long two point connection aluminum extrusion. The connecting beam has four 6 inch aluminum extrusions that line the top and bottom of the letter I which will have suspension connected on both the top and bottom surfaces.

C. Payload Finalized Design

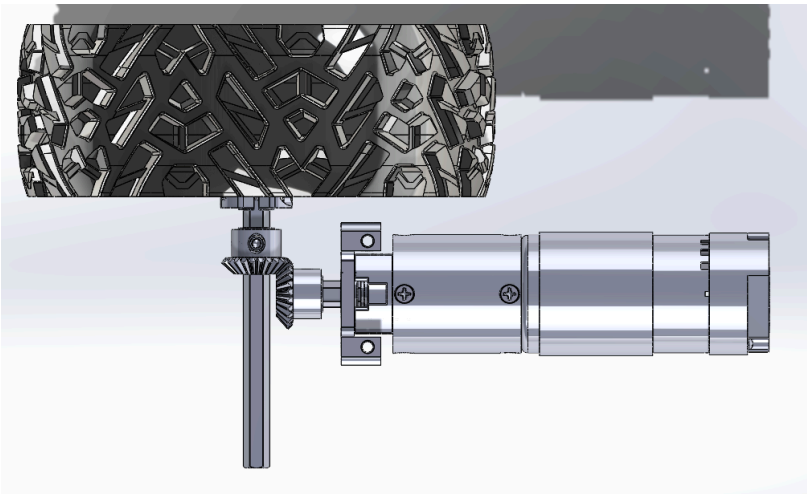


This is the payload design nested within the frame. Corner brackets can be seen connecting the panels of the box together. Near the bottom of the box's side panel, two long brackets can be seen holding up the electronics trays.

D. TPU Spring for Suspension

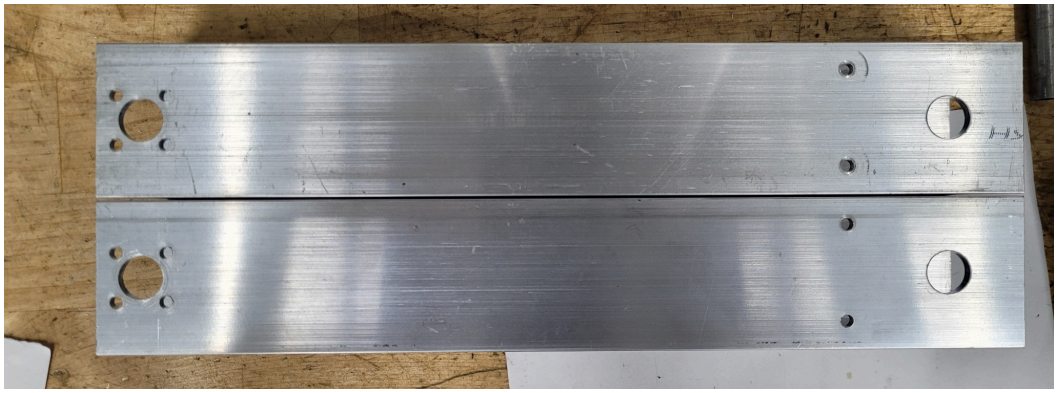
This spring design was printed out of Thermoplastic Polyurethane (TPU), granting it an elasticity comparable to that of a spring. Its bottom face includes a rail that will allow it to slide into the T-slot extrusion.

E. Internals of Arm’s Motion Transfer



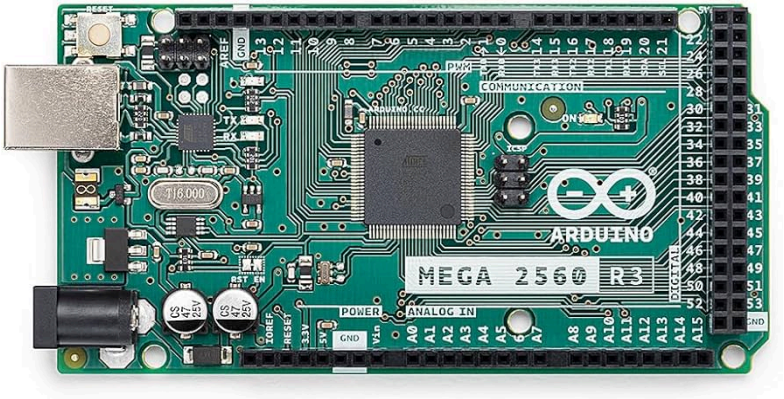
The drive motor will be placed perpendicular to the desired rotation and will be translated using a bevel gear.

F. Machined Aluminum Arms



G. Assembled Arm

H. Arduino Mega Microcontroller



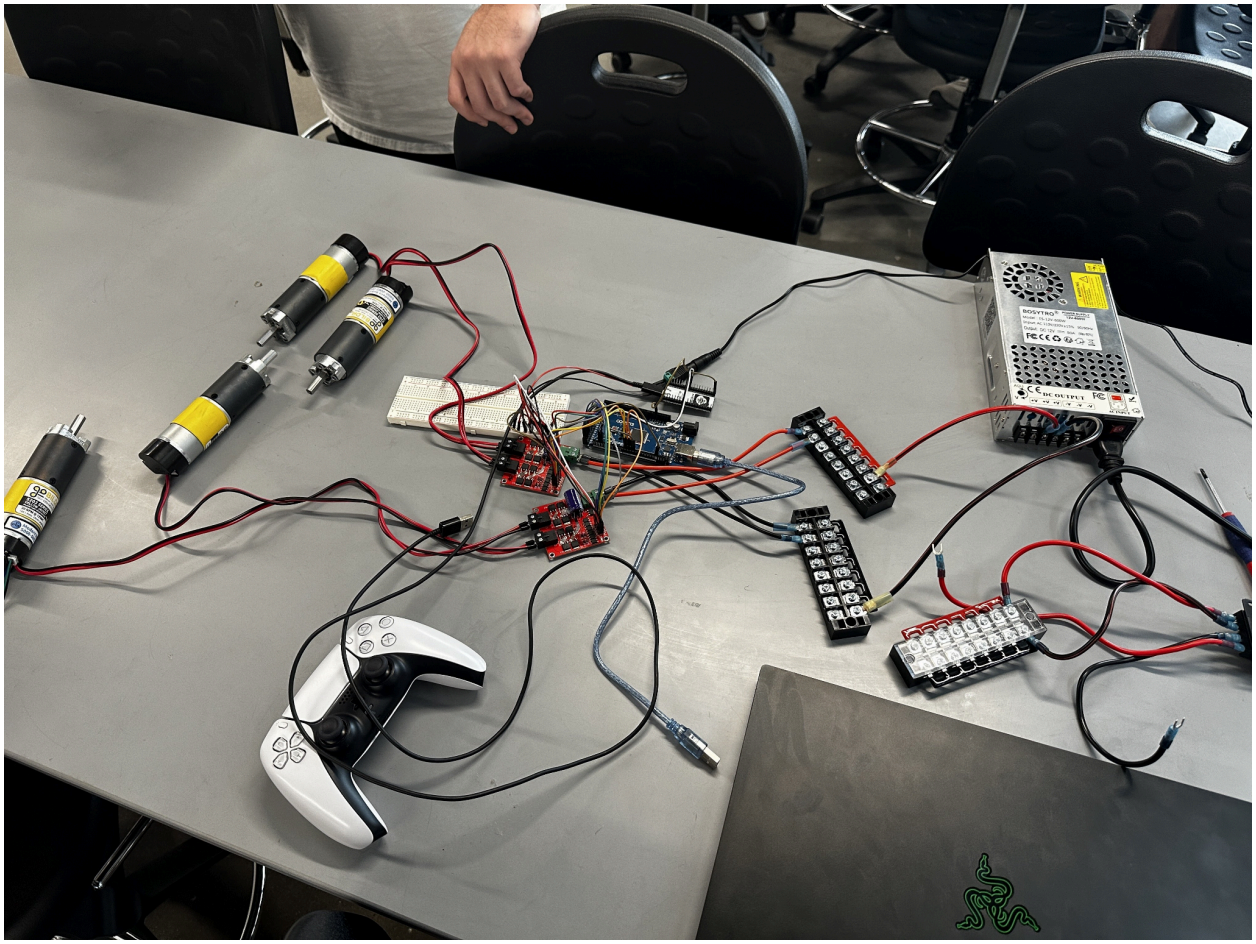
The arduino mega microcontroller will be the logic in our prototype. It will be powered by a 12 volt power supply.

I. Budget Report

Item #	Item	Quantity	Price
1	goBILDA Yellow Jacket Planetary Gear Motor	6	$44.99
2	DROK DC Motor Driver, L298 Dual H Bridge Motor Speed Controller	3	$15.99
3	ELEGOO MEGA R3 Board ATmega	1	$22.99
4	SparkFun 6 Degrees of Freedom Breakout - LSM6DSO	1	$13.95
5	Ultrasonic Sensor	1	$6.99
6	2-Pack 12V to 5V Converter	2	$14.99
7	3-Wire Appliance and Power Tool Cord, 9ft, 16AWG, 13A/125V AC	1	$7.57
8	goBILDA 2317 Series MOD 0.8 Steel Miter Gear	12	$9.99
9	14 AWG Wire	1	$9.49
10	8 AWG Wire	1	$17.99
11	12V DC Power Supply 600W 50A AC to DC Converter Switching Power Supply	1	$36.00
12	goBILDA 1611 Series Flanged Ball Bearing	6	$5.99
13	8 Positions Dual Row 600V 25A Screw Terminal Strip Blocks	1	$14.99

14	goBILDA 1201 Series Quad Block Pattern Mount	6	$6.99
15	1600lbs Hand Winch, 32ft	1	$31.99
16	RC Light Bar 30 LED	1	$15.99
17	Low-Carbon Steel Hex Bar (6ft)	1	$13.64
18	Rivet for End Caps	2	$3.17
19	10 Sets Heavy Duty Aluminum Extrusion Corner Brackets	1	$21.66
20	2x1in T-Slot Framing (6in long)	4	$8.08
21	2x2in T-Slot Framing (3ft long)	1	$42.94
22	2x2in Aluminum Rectangular Tubing (2ft long)	2	$38.77
23	200pcs M5 T-Nuts Screws Kit	1	$13.99
24	10 Pcs Cross Flat Plates for 2020 Aluminum Extrusion	1	$7.99
25	20Pcs 2020 Corner Bracket Joint Plate T Shape	1	$14.99
26	6061 Aluminum Rectangular Tubes 2"x2" 1/8" thick	4	$22.15
27	goBILDA 1310 Series Hyper Hub	12	$7.99
28	2x1in T-Slot Framing (2ft long)	1	$18.39

J. In-Progress Electronics Hardware (12V system complete; 5V in-progress)



K. ESP32 Code v1. (Interpretation completed; feedback in-progress)

```
//Serial Communication
#define TXD_PIN 17
#define RXD_PIN 16

#include <Bluepad32.h>

ControllerPtr myControllers[BP32_MAX_GAMEPADS];

// This callback gets called any time a new gamepad is connected.
// Up to 4 gamepads can be connected at the same time.
void onConnectedController(ControllerPtr ctl) {
    bool foundEmptySlot = false;
    for (int i = 0; i < BP32_MAX_GAMEPADS; i++) {
```

```
            if (myControllers[i] == nullptr) {
                Serial.printf("CALLBACK: Controller is connected,
index=%d\n", i);
                // Additionally, you can get certain gamepad properties
like:
                // Model, VID, PID, BTAddr, flags, etc.
                ControllerProperties properties = ctl->getProperties();
                Serial.printf("Controller model: %s, VID=0x%04x,
PID=0x%04x\n", ctl->getModelName().c_str(), properties.vendor_id,
                              properties.product_id);
                myControllers[i] = ctl;
                foundEmptySlot = true;
                break;
            }
        }
        if (!foundEmptySlot) {
            Serial.println("CALLBACK: Controller connected, but could
not found empty slot");
        }
    }

    void onDisconnectedController(ControllerPtr ctl) {
        bool foundController = false;

        for (int i = 0; i < BP32_MAX_GAMEPADS; i++) {
            if (myControllers[i] == ctl) {
                Serial.printf("CALLBACK: Controller disconnected from
index=%d\n", i);
                myControllers[i] = nullptr;
                foundController = true;
                break;
            }
        }

        if (!foundController) {
            Serial.println("CALLBACK: Controller disconnected, but not
found in myControllers");
        }
    }
```

```
    void dumpGamepad(ControllerPtr ctl) {
        Serial2.printf(
            "idx=%d, dpad: 0x%02x, buttons: 0x%04x, axis L: %4d, %4d,
axis R: %4d, %4d, brake: %4d, throttle: %4d, "
            "misc: 0x%02x, gyro x:%6d y:%6d z:%6d, accel x:%6d y:%6d
z:%6d\n",
            ctl->index(),        // Controller Index
            ctl->dpad(),         // D-pad
            ctl->buttons(),      // bitmask of pressed buttons
            ctl->axisX(),        // (-511 - 512) left X Axis
            ctl->axisY(),        // (-511 - 512) left Y axis
            ctl->axisRX(),       // (-511 - 512) right X axis
            ctl->axisRY(),       // (-511 - 512) right Y axis
            ctl->brake(),        // (0 - 1023): brake button
            ctl->throttle(),     // (0 - 1023): throttle (AKA gas)
button
            ctl->miscButtons(),  // bitmask of pressed "misc" buttons
            ctl->gyroX(),        // Gyro X
            ctl->gyroY(),        // Gyro Y
            ctl->gyroZ(),        // Gyro Z
            ctl->accelX(),       // Accelerometer X
            ctl->accelY(),       // Accelerometer Y
            ctl->accelZ()        // Accelerometer Z
        );
    }

    void dumpMouse(ControllerPtr ctl) {
        Serial.printf("idx=%d, buttons: 0x%04x, scrollWheel=0x%04x,
delta X: %4d, delta Y: %4d\n",
                      ctl->index(),        // Controller Index
                      ctl->buttons(),      // bitmask of pressed
buttons
                      ctl->scrollWheel(),  // Scroll Wheel
                      ctl->deltaX(),       // (-511 - 512) left X Axis
                      ctl->deltaY()        // (-511 - 512) left Y axis
        );
    }

    void dumpKeyboard(ControllerPtr ctl) {
        static const char* key_names[] = {
```

```
            // clang-format off
            // To avoid having too much noise in this file, only a few
keys are mapped to strings.
            // Starts with "A", which is offset 4.
            "A", "B", "C", "D", "E", "F", "G", "H", "I", "J", "K", "L",
"M", "N", "O", "P", "Q", "R", "S", "T", "U", "V",
            "W", "X", "Y", "Z", "1", "2", "3", "4", "5", "6", "7", "8",
"9", "0",
            // Special keys
            "Enter", "Escape", "Backspace", "Tab", "Spacebar",
"Underscore", "Equal", "OpenBracket", "CloseBracket",
            "Backslash", "Tilde", "SemiColon", "Quote", "GraveAccent",
"Comma", "Dot", "Slash", "CapsLock",
            // Function keys
            "F1", "F2", "F3", "F4", "F5", "F6", "F7", "F8", "F9", "F10",
"F11", "F12",
            // Cursors and others
            "PrintScreen", "ScrollLock", "Pause", "Insert", "Home",
"PageUp", "Delete", "End", "PageDown",
            "RightArrow", "LeftArrow", "DownArrow", "UpArrow",
            // clang-format on
        };
        static const char* modifier_names[] = {
            // clang-format off
            // From 0xe0 to 0xe7
            "Left Control", "Left Shift", "Left Alt", "Left Meta",
            "Right Control", "Right Shift", "Right Alt", "Right Meta",
            // clang-format on
        };
        Serial.printf("idx=%d, Pressed keys: ", ctl->index());
        for (int key = Keyboard_A; key <= Keyboard_UpArrow; key++) {
            if (ctl->isKeyPressed(static_cast<KeyboardKey>(key))) {
                const char* keyName = key_names[key-4];
                Serial.printf("%s,", keyName);
           }
        }
        for (int key = Keyboard_LeftControl; key <= Keyboard_RightMeta;
key++) {
            if (ctl->isKeyPressed(static_cast<KeyboardKey>(key))) {
                const char* keyName = modifier_names[key-0xe0];
```

```
                Serial.printf("%s,", keyName);
            }
        }
        Console.printf("\n");
    }

    void dumpBalanceBoard(ControllerPtr ctl) {
        Serial.printf("idx=%d,  TL=%u, TR=%u, BL=%u, BR=%u,
temperature=%d\n",
                      ctl->index(),        // Controller Index
                      ctl->topLeft(),      // top-left scale
                      ctl->topRight(),     // top-right scale
                      ctl->bottomLeft(),   // bottom-left scale
                      ctl->bottomRight(),  // bottom-right scale
                      ctl->temperature()   // temperature: used to
adjust the scale value's precision
        );
    }

    void processGamepad(ControllerPtr ctl) {
        // There are different ways to query whether a button is
pressed.
        // By query each button individually:
        //  a(), b(), x(), y(), l1(), etc...
        if (ctl->a()) {
            static int colorIdx = 0;
            // Some gamepads like DS4 and DualSense support changing the
color LED.
            // It is possible to change it by calling:
            switch (colorIdx % 3) {
                case 0:
                    // Red
                    ctl->setColorLED(255, 0, 0);
                    break;
                case 1:
                    // Green
                    ctl->setColorLED(0, 255, 0);
                    break;
                case 2:
                    // Blue
```

```
                    ctl->setColorLED(0, 0, 255);
                    break;
                }
                colorIdx++;
            }

            if (ctl->b()) {
                // Turn on the 4 LED. Each bit represents one LED.
                static int led = 0;
                led++;
                // Some gamepads like the DS3, DualSense, Nintendo Wii,
Nintendo Switch
                // support changing the "Player LEDs": those 4 LEDs that
usually indicate
                // the "gamepad seat".
                // It is possible to change them by calling:
                ctl->setPlayerLEDs(led & 0x0f);
            }

            if (ctl->x()) {
                // Some gamepads like DS3, DS4, DualSense, Switch, Xbox One
S, Stadia support rumble.
                // It is possible to set it by calling:
                // Some controllers have two motors: "strong motor", "weak
motor".
                // It is possible to control them independently.
                ctl->playDualRumble(0 /* delayedStartMs */, 250 /*
durationMs */, 0x80 /* weakMagnitude */,
                                    0x40 /* strongMagnitude */);
            }

            // Another way to query controller data is by getting the
buttons() function.
            // See how the different "dump*" functions dump the Controller
info.
            dumpGamepad(ctl);
        }

        void processMouse(ControllerPtr ctl) {
            // This is just an example.
```

```
    if (ctl->scrollWheel() > 0) {
        // Do Something
    } else if (ctl->scrollWheel() < 0) {
        // Do something else
    }

    // See "dumpMouse" for possible things to query.
    dumpMouse(ctl);
}

void processKeyboard(ControllerPtr ctl) {
    if (!ctl->isAnyKeyPressed())
        return;

    // This is just an example.
    if (ctl->isKeyPressed(Keyboard_A)) {
        // Do Something
        Serial.println("Key 'A' pressed");
    }

    // Don't do "else" here.
    // Multiple keys can be pressed at the same time.
    if (ctl->isKeyPressed(Keyboard_LeftShift)) {
        // Do something else
        Serial.println("Key 'LEFT SHIFT' pressed");
    }

    // Don't do "else" here.
    // Multiple keys can be pressed at the same time.
    if (ctl->isKeyPressed(Keyboard_LeftArrow)) {
        // Do something else
        Serial.println("Key 'Left Arrow' pressed");
    }

    // See "dumpKeyboard" for possible things to query.
    dumpKeyboard(ctl);
}

void processBalanceBoard(ControllerPtr ctl) {
    // This is just an example.
```

```
            if (ctl->topLeft() > 10000) {
                // Do Something
            }

            // See "dumpBalanceBoard" for possible things to query.
            dumpBalanceBoard(ctl);
        }

        void processControllers() {
            for (auto myController : myControllers) {
                if (myController && myController->isConnected() &&
myController->hasData()) {
                    if (myController->isGamepad()) {
                        processGamepad(myController);
                    } else if (myController->isMouse()) {
                        processMouse(myController);
                    } else if (myController->isKeyboard()) {
                        processKeyboard(myController);
                    } else if (myController->isBalanceBoard()) {
                        processBalanceBoard(myController);
                    } else {
                        Serial.println("Unsupported controller");
                    }
                }
            }
        }

        // Arduino setup function. Runs in CPU 1
        void setup() {
            Serial.begin(115200);   // USB Serial (for debugging)
            Serial2.begin(9600, SERIAL_8N1, RXD_PIN, TXD_PIN);  // UART2
(ESP32 <-> Mega)

            Serial.printf("Firmware: %s\n", BP32.firmwareVersion());
            const uint8_t* addr = BP32.localBdAddress();
            Serial.printf("BD Addr: %2X:%2X:%2X:%2X:%2X:%2X\n", addr[0],
addr[1], addr[2], addr[3], addr[4], addr[5]);

            // Setup the Bluepad32 callbacks
            BP32.setup(&onConnectedController, &onDisconnectedController);
```

```
        // "forgetBluetoothKeys()" should be called when the user
performs
        // a "device factory reset", or similar.
        // Calling "forgetBluetoothKeys" in setup() just as an example.
        // Forgetting Bluetooth keys prevents "paired" gamepads to
reconnect.
        // But it might also fix some connection / re-connection issues.
        BP32.forgetBluetoothKeys();

        // Enables mouse / touchpad support for gamepads that support
them.
        // When enabled, controllers like DualSense and DualShock4
generate two connected devices:
        // - First one: the gamepad
        // - Second one, which is a "virtual device", is a mouse.
        // By default, it is disabled.
        BP32.enableVirtualDevice(false);
    }

    // Arduino loop function. Runs in CPU 1.
    void loop() {
        // This call fetches all the controllers' data.
        // Call this function in your main loop.
        bool dataUpdated = BP32.update();
        if (dataUpdated)
            processControllers();

        // The main loop must have some kind of "yield to lower priority
task" event.
        // Otherwise, the watchdog will get triggered.
        // If your main loop doesn't have one, just add a simple
`vTaskDelay(1)`.
        // Detailed info here:
        //
https://stackoverflow.com/questions/66278271/task-watchdog-got-triggered-t
he-tasks-did-not-reset-the-watchdog-in-time

        //     vTaskDelay(1);
        delay(150);
```

```
}
```

Arduino Mega Code v1. (PID integration in progress)

```
#include <stdio.h>
#include <string.h>
#include <UnorderedMap.h>
#include <StringAction.h>

// Motor PWM Assignments
#define setFL 3
#define setFR 4
#define setBL 5
#define setBR 6
#define setFA 7
#define setBA 8

//Motor Digital Assignments
#define FL1 22  // Front Left Wheel
#define FL2 23
#define FR1 24  // Front Right Wheel
#define FR2 25
#define BL1 26  // Back Left Wheel
#define BL2 27
#define BR1 28  // Back Right Wheel
#define BR2 29
#define FA1 30  // Front Arm
#define FA2 32
#define BA1 33  // Back Arm
#define BA2 34

//Encoder Digital Pins
#define encFA_A 36 // Front Arm Encoder
#define encFA_B 37
```

```
#define encBA_A 38 // Back Arm Encoder
#define encBA_B 39


//Empty Array for values to be read from the ESP32
UnorderedMap<String, float> dataValues;
String dataKeys[] = {"dpad", "buttons", "axisLX", "axisLY",
"axisRX", "axisRY", "brake", "throttle"};
UnorderedMap<String, int> keyIndices;


void setup() {
    // Set indices for each key in hashmap
    keyIndices.put("dpad", 0);
    keyIndices.put("buttons", 1);
    keyIndices.put("axisLX", 2);
    keyIndices.put("axisLY", 3);
    keyIndices.put("axisRX", 4);
    keyIndices.put("axisRY", 5);
    keyIndices.put("brake", 6);
    keyIndices.put("throttle", 7);

    //Start Serial Comms
    Serial.begin(115200);
    Serial1.begin(9600);

    //Pin Initialization

    // PWM Motor Pins
    pinMode(setFL, OUTPUT);
    pinMode(setFR, OUTPUT);
    pinMode(setBL, OUTPUT);
    pinMode(setBR, OUTPUT);
    pinMode(setFA, OUTPUT);
    pinMode(setBA, OUTPUT);

    // Digital Motor Pins
    pinMode(FL1, OUTPUT);
    pinMode(FL2, OUTPUT);
    pinMode(FR1, OUTPUT);
```

```
        pinMode(FR2, OUTPUT);
        pinMode(BL1, OUTPUT);
        pinMode(BL2, OUTPUT);
        pinMode(BR1, OUTPUT);
        pinMode(BR2, OUTPUT);
        pinMode(FA1, OUTPUT);
        pinMode(FA2, OUTPUT);
        pinMode(BA1, OUTPUT);
        pinMode(BA2, OUTPUT);

        //Encoders
        pinMode(encFA_A, INPUT); // Front Arm Encoder
        pinMode(encFA_B, INPUT);
        pinMode(encBA_A, INPUT); // Rear Arm Encoder
        pinMode(encBA_B, INPUT);

        // attachInterrupt(digitalPinToInterrupt(encFA_A),readEncoder,
RISING);
        // attachInterrupt(digitalPinToInterrupt(encBA_A),readEncoder,
RISING);
    }

    void loop() {


        if (Serial1.available()) {
            String data = Serial1.readStringUntil('\n');
            data.trim();
            Serial.println(data);
            dataParser(data);
            //printValues();
        }
        float combo = customValue("throttle") - customValue("brake");
        if (combo < 0) {
          combo = 0;
        }
        float* vectors = motorVectors(combo);

        //Drive Motors
        driveMotor("FL", vectors[0]);
```

```
          driveMotor("FR", vectors[1]);
          driveMotor("BL", vectors[2]);
          driveMotor("BR", vectors[3]);

          // //Show all PWM values
          // Serial.print(String(combo));
          // Serial.print(" ");  // Space between values

          // for (int i = 0; i < 4; i++) {  // Assuming coeffs has 4
elements
          // Serial.print(vectors[i]);
          // Serial.print(" ");  // Space between values
          // }
          // Serial.println();  // Move to a new line

          // Drive Motor




      }

      void dataParser(String data){
        StringAction s;
        String split_arr[12];
        s.split(data, split_arr, ",");

        for (int i = 1; i < 9; i++) {
          String value = split_arr[i];
          value.replace(" ", "");
          value.replace("0x", "");
          int index = value.indexOf(":");
          if (index > -1) {
            value = value.substring(index + 1);
          }
          dataValues.put(dataKeys[i-1], value.toFloat()); // TODO: hex
values?? 0x000a
        }
```

```
}

void printValues() {
  Serial.println("\n\n************ PRINTING VALUES ************");
  for (String key: dataKeys) {
    Serial.println(key + " = " + String(dataValues.getValue(key)));
  }

  Serial.println();
}

float customValue(String param) {
  int index = keyIndices.getValue(param);
  float value = dataValues.getValue(param);
  switch (index) {
    case 0: // dpad
      break;
    case 1: // buttons
      break;
    case 2: // axisLX
      break;
    case 3: // axisLY
      break;
    case 4: // axisRX
      break;
    case 5: // axisRY
      break;
    case 6: // brake
      return value/4;
    case 7: // throttle
      return value/4;
    default:
      return -1;
  }
  return value;
}

float* motorVectors(float speed) {
    static float coeffs[4]; // Static array to persist after
function return
```

```
            float x = customValue("axisLX"); // Get joystick X-axis value
            float limiter = 14.0 / 15.0; // Steering sensitivity limiter
            float btnState = customValue("buttons");

            // Define deadzone threshold
            const int deadzone = 30;

            // Normalize `x` between -1.0 (full left) to 1.0 (full right),
with deadzone
            if (abs(x) < deadzone) x = 0; // Deadzone effect
            float normX = x / 512.0; // Assuming raw input ranges from -512
to 512
            if (normX > 1.0) normX = 1.0;
            if (normX < -1.0) normX = -1.0;

            // Base speed factor for straight movement
            float baseSpeed = speed * limiter;

            // Default multipliers
            float leftMultiplier = 1.0;
            float rightMultiplier = 1.0;

            // Handle sharp turns when joystick is near the extremes
            if (normX <= -0.9) {  // Extreme left turn → Left wheels go
backward
                leftMultiplier = -1.0;
                rightMultiplier = 1.0;
            } else if (normX >= 0.9) {  // Extreme right turn → Right wheels
go backward
                leftMultiplier = 1.0;
                rightMultiplier = -1.0;
            } else {
                // Gradual Steering Control for slight turns
                leftMultiplier = (normX < 0) ? (1.0 + normX) : 1.0;  //
Reduce left side speed for left turns
                rightMultiplier = (normX > 0) ? (1.0 - normX) : 1.0; //
Reduce right side speed for right turns
            }
            if (btnState == 82){
```

```
            leftMultiplier = leftMultiplier * -1;
            rightMultiplier = rightMultiplier * -1;
        }

        // Assign motor values
        coeffs[0] = baseSpeed * leftMultiplier;  // Front Left
        coeffs[1] = baseSpeed * leftMultiplier;  // Back Left
        coeffs[2] = baseSpeed * rightMultiplier; // Front Right
        coeffs[3] = baseSpeed * rightMultiplier; // Back Right

        return coeffs;
    }

    void driveMotor(String motor, float vector) {
        int pwmPin, dirPin1, dirPin2;

        if (motor == "FL") { pwmPin = setFL; dirPin1 = FL1; dirPin2 =
FL2; }
        else if (motor == "FR") { pwmPin = setFR; dirPin1 = FR1; dirPin2
= FR2; }
        else if (motor == "BL") { pwmPin = setBL; dirPin1 = BL1; dirPin2
= BL2; }
        else if (motor == "BR") { pwmPin = setBR; dirPin1 = BR1; dirPin2
= BR2; }
        else return; // Ignore invalid motor names

        if (vector == 0) {
            analogWrite(pwmPin, 0);
            digitalWrite(dirPin1, LOW);
            digitalWrite(dirPin2, LOW);
            Serial.println(motor + ": Stopped");
        } else if (vector > 0) {
            analogWrite(pwmPin, abs(vector));
            digitalWrite(dirPin1, HIGH);
            digitalWrite(dirPin2, LOW);
            Serial.println(motor + ": Forward " + String(abs(vector)));
        } else {
            analogWrite(pwmPin, abs(vector));
            digitalWrite(dirPin1, LOW);
            digitalWrite(dirPin2, HIGH);
```

```
        Serial.println(motor + ": Reverse " + String(abs(vector)));
    }
}


// void readEncoder(){
//   int b = digitalRead(enc1B);
//   if(b>0){
//     pos++;
//   }
//   else{
//     pos--;
//   }


// }
```