

Engineering Manual

WRO

Team The Redacted

August 25, 2026

Contents

1 Selection of Components **5**
- 1.1 Sensing . . . 5
- 1.2 Compute . . . 7
- 1.3 Motion . . . 8
  - 1.3.1 Drive System . . . 8
  - 1.3.2 Steering System . . . 9
- 1.4 Power Management . . . 10
- 1.5 The Additional Hardware . . . 12
  - 1.5.1 Bearings . . . 12
  - 1.5.2 Carbon Fiber Tubes . . . 14
- 1.6 Wheels . . . 16

2 PCB Assembly **18**
- 2.1 Purpose . . . 18
- 2.2 Form Factor . . . 19
- 2.3 Selection of the ICs . . . 19
  - 2.3.1 The 5.1V Rail . . . 19
  - 2.3.2 The 8V Rail . . . 20
- 2.4 The Schematic . . . 20
- 2.5 The Layout & Routing . . . 23
- 2.6 Assembly . . . 28
  - 2.6.1 Advice . . . 29
- 2.7 Mistakes & Errors . . . 30

3 3D Prints & CAD **32**
- 3.1 Layout & Wheelbase . . . 32
- 3.2 The Steering Assembly . . . 34
- 3.3 The Differential . . . 39
- 3.4 The Rear Axle . . . 44
- 3.5 The Center . . . 47
- 3.6 Accessories . . . 51
- 3.7 Tools . . . 54
- 3.8 Notes . . . 57

4 OS and Configurations **58**
- 4.1 Camera ROS 2 Integration . . . 58
- 4.2 Hardware PWM Configuration & Tuning . . . 59

5 Control Architecture & Software System **60**
- 5.1 System Architecture & ROS 2 Pipeline . . . 60
- 5.2 ROS 2 Topics & Interfaces . . . 60
- 5.3 Perception Algorithms & Path Extraction . . . 61
  - 5.3.1 RANSAC Wall Extraction & Heading Offset . . . 61
  - 5.3.2 Dynamic Velocity-Dependent Braking . . . 61
- 5.4 Challenge Execution Logic . . . 61
  - 5.4.1 Round One: Obstacle-Free Wall Following . . . 61
  - 5.4.2 Round Two: Pillar Mapping & Parallel Parking . . . 61

6 Code Architecture **63**
6.1 Round 1: Obstacle-Free Track Traversal 63
6.2 Round 2: Obstacle Avoidance & Vision-LiDAR Fusion 65

7 Conclusion & Acknowledgments **69**
7.1 Document Production 69
7.2 Open-Source Acknowledgments 69

List of Figures

1 The YD-LiDAR X4 Pro. . . . 6
2 The IMX219 with a very wide angle lens. . . . 6
3 The Raspberry Pi 4 Model B. . . . 7
4 The Motor. . . . 8
5 The ESC. . . . 9
6 The Configuration Device that connects via PWM. . . . 9
7 The S2500M metal gear servo motor. . . . 10
8 The DOGCOM 4s 850 mAh battery. . . . 12
9 The internals of a Combined needle roller / thrust ball bearing. . . . 13
10 The 6901ZZ roller ball bearing. . . . 13
11 The Needle Roller bearing. . . . 14
12 The 12 mm CF rod. . . . 15
13 The 3 mm PultrudedCF rod. . . . 15
14 The LM61460 in the VQFN-HR package. . . . 20
15 The TPS62932 in the SOT-583 package. . . . 20
16 The files for the schematic given by TI Workbench. . . . 21
17 The 3 pages of the schematic. . . . 22
18 The "edge cuts" layer. . . . 23
19 The fixed items layer. . . . 24
20 The final layout depicting the placement of all the components. . . . 25
21 The 4 copper layers. Layer1 $\rightarrow$ 4. . . . 26
22 The rendered PCB. . . . 27
23 The solder paste on the pads after being applied with a stencil. . . . 28
24 The Interactive HTML BOM Tool. . . . 29
25 The Bodge Wire. . . . 31
26 The Onshape assembly. . . . 33
27 The Front Wheel Adapter Pin. . . . 34
28 The Steering Knuckle. . . . 35
29 The Steering Column. . . . 35
30 The Top Brace. . . . 36
31 The Bottom Brace. . . . 36
32 The Side Brace. . . . 37
33 The Servo Horn. . . . 37
34 The Steering Links 38
35 The TPU Spacer. . . . 39
36 The Differential Gearbox. . . . 39
37 The Differential Gears 40
38 The Spider Center. . . . 41
39 The Housing for the Gearbox. . . . 42
40 The Attaching Cup. . . . 43
41 The Spacer Disk. . . . 44
42 The Rear Axle Assembly. . . . 45
43 The Front Mount. . . . 46
44 The Front Mount in the Slicer. . . . 46
45 The Back Mount. . . . 47
46 The Center Assembly. . . . 48

47 The Top Brace. 49
48 The Bottom Left and Right Brace. 50
49 The Left and Right Bearing mounts. 50
50 The Lidar & ESC mount. 51
51 The Wheel Hub. 52
52 The Raspberry Pi Mount. 53
53 The ESC Wire Mount. 53
54 The Mold for the Polyurethane Tire. 54
55 The Standoff Cutting Tool. 55
56 The Axle Cutting Tool. 56
57 The Axle Drilling Tool. 57
58 ROS 2 System Control Architecture and Inter-Node Dataflow Diagram. 60
59 Round 1 Finite State Machine (FSM) Execution Flow. 65
60 Round 2 Perception, Sensor Fusion, and Reactive Control Dataflow Pipeline. . . 66

1 Selection of Components

1.1 Sensing

The WRO Future Engineers challenge sets certain criteria on the needed hardware. The second challenge makes teams not just detect walls but the pillars and their colors, as well as the ability to parallel park.

This means that a comprehensive and accurate inference of the playing field is required. There are multiple paths one can go down to fulfill this, there are perhaps only two practical routes. One is using distance sensors along with a camera and the other is using a 2-dimensional LiDAR.

The trade-offs between the two are fairly significant. The LiDAR comes at a higher cost than the camera and provides absolute and accurate cloud point data that can be used to detect the pillars and construct a map of the playing field with high accuracy. While the camera system is cheaper in the hardware cost, it has a significant cost in terms of compute. The detection of the actual pillars in the second round is easy, disregarding how slowly it would have to run and train (about $1/2 - 1$ Hz for good reliability on a wide FOV camera running YOLO, taking about 3 hours to train on 300 images, and a variable time to compute, not real time), lacks in the accuracy of positioning the pillars in dynamic environments. The LiDAR also provides very accurate and fast ($8 - 20$Hz) feedback for absolute positioning on the go, something the camera does not. The aspect of parallel parking also plays into the advantage of the LiDAR. The LiDAR can provide a lot more detail than a distance assisted vision system could provide. If one did not use a LiDAR, atleast 4 mutually perpendicular distance sensor, Time of Flight or Ultrasonic, would have to be implemented on an I2C bus.

All this made the case for using the LiDAR stronger. On smaller scales, where the camera is quite a bit lager than the vehicle and so is the LiDAR, using a LiDAR makes sense due to the current lack of compute density for very small and light boards being an issue. But, for larger vehicles, where the cost of cameras scales linearly and the LiDAR scales exponentially, using multiple cameras and cheaper, larger compute makes sense. Hence, we decided to use a LiDAR for our vehicle.

However, this did come at a major trade-off and design cost. We had to keep the whole vehicle shorter than 80 mm to accommodate for a clear view of the laser at about 90 mm and have a clearance of 10 mm from the top of the 100 mm arena walls to account for deviations in the walls and potentially the playing field as well.

For the actual hardware, the **YD-LiDAR X4 Pro** was chosen due to it's compatibility with ROS and it's relatively low cost and high scanning speed along with it's resolution. The X4 Pro provides the high range that is needed, extending from 8 centimeters to 10 meters. When measuring less than 8 centimeters, parallel parking for example, the fall further away will be utilized to interpolate. The USB interface of the LiDAR also was very handy along with the openly available library and drivers.



Figure 1: The YD-LiDAR X4 Pro.

Because the LiDAR only outputs the point cloud data, and we need to find the color of the pillars as well, a wide FOV camera is used in tandem. It is only used to determine the color of the pillar and not used for detection. The LiDAR detects the pillar and the image from the camera is then used to assign the color. This keeps the compute costs very low and allows for very fast inference. The camera used is the **Arducam IMX219 Wide Angle Camera**. It has a field of view of 175°. This allows us to have the maximum window for inference possible as we don't need the data resolution a camera based solution would.

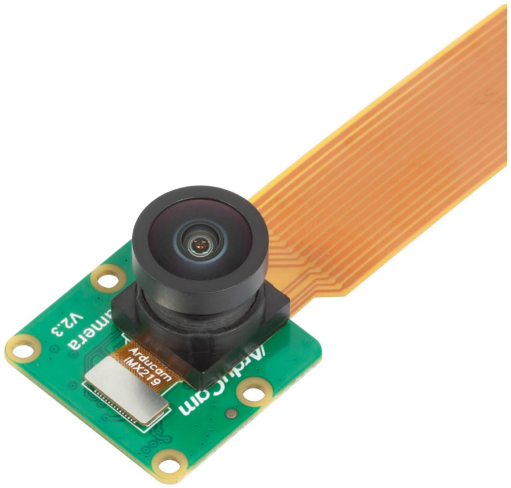


Figure 2: The IMX219 with a very wide angle lens.

1.2 Compute

The compute module doesn't need to be something excessive like the Jetson Orion Nano. The compute required to complete both the challenges is fairly low and could be done on an STM32 H7 micro-controller as well. The compute module has to fulfill the following requirements:

1. Small form factor.
2. Have a USB poet.
3. Have a Camera Serial Interface .
4. Run ROS or Micro ROS.
5. Have a decent number of exposed IO pins for accessories.
6. Have the ability to read/write code for easy iterations.

This led to the Raspberry Pi 4B. Any other similar Single Board Computer (SBC) can be used. The 4GB RAM version was chosen so that a graphical user interface could be ran on it if needed when editing code during the competition. It would also provide sufficient headroom. A 2GB RAM version would have also been sufficient. A Raspberry Pi 5 would be overkill. The Pi 4 is also cheaper and easier to source. It would also have the compute needed to run the control loop at higher frequencies and have faster and more accurate control.

The only downside of the Rspberry Pi was it's size. It's a lot larger than needed, and occupies crucial space in the limited volume. This was solved with intentional and clever design. It's availability of USB ports and the CSI makes it a very appealing solution. Custom hardware, ST based micro-controller or a board with the Raspberry Pi Compute Module, can be used but would be harder to source and would involve a longer development time. The Raspberry Pi also acts as a drop in solution as it has an external micro-SD card where all the code is stored, allowing for easy access and replaceability.

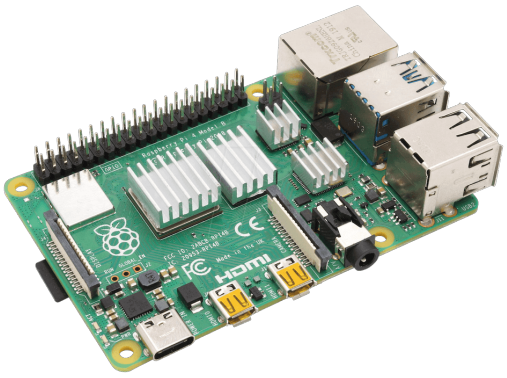


Figure 3: The Raspberry Pi 4 Model B.

1.3 Motion

A **rear wheel drive system** was chosen as it would distribute the complexity of the steering and drive systems, allowing easier manufacturing and assembly. The trade-off made here was that a **differential would be required** to prevent under steer that would be cause when turning at tight radii and slow speeds. This increases the complexity slightly, but not as much as using a drive and steering would need.

The motion system can be broken down into two parts, the drive system and the steering system.

1.3.1 Drive System

A brush less DC (BLDC) motor is proffered over a brushed motor due to it's higher resolution of speed changes particularly at a low speeds. That comes particularly handy when parallel parking. It is also preferred due to its efficiency and torque. The ideal case would be a stepper or BLDC motor with Field Oriented Control (FOC) as it would provide excellent position and torque control, but sourcing such FOC motor drivers in a small and compact size is very hard and expensive. A note should be made that developing such a custom FOC motor driver for 10-20A should not be hard but would take time.

For the motor, a Surpass Hobby **36 mm diameter in runner BLDC motor** is used. The 36 mm diameter is chosen as it provides enough **torque, acceleration and internal breaking** for a 1.5 kg vehicle. The length of the motor is 74 mm. This makes it a **3674 motor**. The motor is also widely used in scale RC cars and hence is easy to source. The motor is rated at **1700 kv**, which provides generous speed and high torque.



Figure 4: The Motor.

A Surpass Hobby 3 Phase **120A 4S ESC** is used. This Electronic Speed Controller is pre-

ferred due to its high **reliability and headroom, along with its configurability**. The ESC can be configured to operate at certain speed, torque and braking levels with a configurator device.



Figure 5: The ESC.

Do note that if built again, a shorter motor and lower ampere rated ESC would be used, something like a 3650 motor and a 60A ESC. The 3674 motor is too long and has excessive torque that is not really needed. The shorter motor should be used to reduce the weight as much as possible. The ideal motor would be a 3650 motor rated somewhere about 1000 kv. The 60A ESC would also be chosen to reduce weight. The current setup was chosen because it is what was available without purchasing new components.



Figure 6: The Configuration Device that connects via PWM.

1.3.2 Steering System

A **direct drive steering system** was chosen for the vehicle, meaning that the servo directly drives the wheels. This was chosen as the steering links would be manufactured using 3D print-

ing and each joint would cause additional backlash in the linkage mechanism. A system with a grater mechanical leverage would hence be more complex and less accurate, while taking more volume.

The **S2500M** servo motor is used, which provides a torque of about 25 $\frac{kg}{cm}$ at about 8V. This would be provide ample headroom in a direct drive steering system. The only downside to contend with in this system is its weight and the fact that the servo runs on 8V, meaning it will need its own power supply. Do note that a lower torque servo that operates on 5V can be used if needed, the S2500M was available without purchasing new components.



Figure 7: The S2500M metal gear servo motor.

1.4 Power Management

The batteries were chosen on strict power consumption metrics.

To provide adequate headroom for the 3+ minutes of operation needed, **2 DOGCOM 4S 850 mAh batteries were used in parallel**. A single 4S 850 mAh battery in testing at full motion capabilities provided power for about 2.5 minutes before it went below the safe discharge voltage of 3.5 V per cell. Hence a second battery is needed for headroom.

The 4S batteries are used as they have the maximum voltage compatible with the setup, allowing the maximum acceleration and torque for the vehicle. The batteries use a **Li-Po (Lithium Polymer)** chemistry as that provides the high discharge current needed. It is also widely available and hence easy to source. On a larger scale vehicle, Li-Ion cells would be used due to their longer discharge cycle rating as the only major draw back with Li-Po batteries are their lower charge cycle rating.

Table 1: System Power Usage and Electrical Specifications

Component	**Voltage Rail**	**Idle Current**	**Typical Draw**	**Peak Draw**	**Power Regulator / Impact**
Raspberry Pi 4B (4GB)	5.15V DC	600 mA (3.1 W)	1.2A–1.5A (6.2–7.7 W)	2.2A–2.5A (11.3–12.9 W)	TI LM61460 (5.15V Buck); Primary SBC running ROS, vision, and control loops.
YD-LiDAR X4 Pro	5.0V / 5.15V DC	280 mA (1.4 W)	330–380 mA (1.7–1.95 W)	800–1000 mA (4.0–5.0 W)	5.15V Rail; 360° 2D LiDAR for mapping, wall-following, and parking.
Arducam IMX219	3.3V / 5.0V DC	100 mA (0.5 W)	150–200 mA (0.75–1.0 W)	300 mA (1.5 W)	RPi CSI Bus; 175° Ultra-Wide FOV camera for pillar color classification.
Surpass Hobby 3674 Motor	14.8V–16.8V DC	2.6A–3.1A (38.5–46 W)	4.0A–7.0A (60–105 W)	15A–25A (220–370 W)	Direct 4S LiPo; 1700KV BLDC rear drivetrain propulsion motor.
Surpass Hobby 120A ESC	14.8V–16.8V DC	100 mA (1.5 W)	100–200 mA (1.5–3.0 W)	120A Cont. / 760A Burst	Direct 4S LiPo; 3-Phase electronic speed controller with PWM interface.
S2500M Servo Motor	8.0V DC	100 mA (0.8 W)	500mA–1.0A (4.0–8.0 W)	1.8A–2.0A (14.4–16.0 W)	TI TPS62932 (8V Buck); 25 kg·cm high-voltage metal gear steering servo.
Peripherals & PCB Logic	5.15V / 8.0V DC	20 mA (0.1 W)	50–100 mA (0.25–0.5 W)	150 mA (0.75 W)	PCB Hat; Onboard status LEDs, switches, and ToF headers.
Total 5.15V Rail	**5.15V DC**	**1.0 A (5.15 W)**	**1.8A–2.1A (9.3–10.8 W)**	**3.5A–3.8A (18.0–19.6 W)**	Regulated by TI LM61460 (6A Cont. Limit).
Total 8.0V Rail	**8.0V DC**	**0.1 A (0.8 W)**	**0.5A–1.0A (4.0–8.0 W)**	**1.8A–2.0A (14.4–16.0 W)**	Regulated by TI TPS62932 (2A Cont. Limit).
Total System Draw	**14.8V Nominal**	**0.8 A (12 W)**	**5.5A–8.5A (80–125 W)**	**18A–28A (260–410 W)**	**2x DOGCOM 4S 850 mAh** (1700 mAh Total).



Figure 8: The DOGCOM 4s 850 mAh battery.

1.5 The Additional Hardware

Additional hardware such as bearings and wheels are needed to build the vehicle. These are things that cannot be 3D printed and are manufactured in other ways or are sourced.

1.5.1 Bearings

For the two front steering knuckles, a **thrust roller bearing** is used. This is a type of bearing that combines a needle and thrust bearing in a single housing. It is used to constraint the location of the two front steering wheels though the axis of rotation and ensure there is no play present. If a normal roller ball bearing would be used, the outward and asymmetric torque of the wheel "pin" would misalign the groove and the balls causing excessive play. The thrust roller bearing is a compact solution used in real cars as well.

The inner diameter of the bearing is 12 mm. It is the "**BCWN-T-d12-IR**" bearing, sourced from JLCMC.

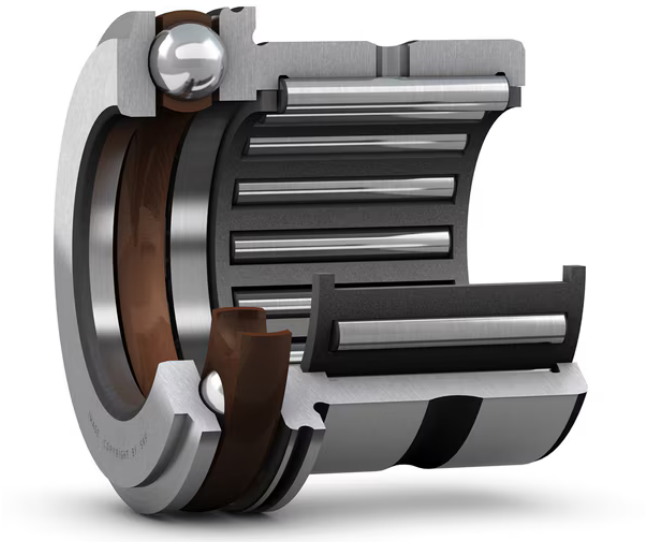


Figure 9: The internals of a Combined needle roller / thrust ball bearing.

For the differential, traditional roller ball bearings were used as the axial forces transferred are not that high. The **6901ZZ** ball bearing is used as it is in a very compact package that has an internal diameter of 12 mm, an external diameter of 24 mm and a width of 6 mm, making it ideal to put in the already compact differential gear box.



Figure 10: The 6901ZZ roller ball bearing.

To support the rear axle, needle roller bearings are used. They are used as they have low friction and can withstand very high forces perpendicular to the axis of rotation, prefect for transferring the weight from the main body of the vehicle to the wheels. The "**BCWG-NK12/12**" bearing was used, sourced from JLCMC. It has an inner diameter of 12 mm, an

outer diameter of 19.5 mm and is 12 mm deep.



Figure 11: The Needle Roller bearing.

1.5.2 Carbon Fiber Tubes

Carbon fiber tubes were used at several places as axles, linkage connectors and structural connectors. This decision was made as carbon fiber has a very high tensile strength and is particularly very stiff perpendicular to it's length.

A combination of 2 diameters were used. The axle for the rear two wheels were made using **12 mm outer diameter** and 10 mm inner diameter carbon fiber tubes. Any aluminum or steel shaft could be used here, but a 3D printed shaft cannot be used as the layer lines of the print would be perpendicular to the layer lines, leading to very weak prints. Carbon fiber was used as it is light weight and relatively easy to source in **roll wrapped rods**. This tube consists of "twill" and not "toe".



Figure 12: The 12 mm CF rod.

The other rod used was a **3 mm outer diameter solid pultruded** rod. It is used in the steering links as well as the structural spin that runs throughout the vehicle to provide stiffness to the frame and not let the printed parts sag. It is pultruded, meaning it does not consist of carbon fibers but of chopped carbon bits held by epoxy. It is comparatively less stiff but still very strong. It consists of "toe" and no "twill".



Figure 13: The 3 mm PultrudedCF rod.

1.6 Wheels

Design Constraints & Diameter Selection

The selection of the wheel diameter is one of the most critical foundational decisions in the vehicle's architecture, as it directly locks in ground clearance, overall vehicle height, wheelbase geometry, and internal volume distribution.

Two strict physical constraints dictated the wheel sizing. First, the maximum overall height of the vehicle had to remain strictly under 80 mm to ensure the YD-LiDAR X4 Pro maintains an unobstructed 360° laser line sweep at a height of 90 mm, which is well clear of the 100 mm arena perimeter walls. Second, the ground clearance had to accommodate the Surpass Hobby 3674 rear drive motor, which has an outer diameter of 36 mm. To prevent the motor housing from dragging against flat ground without suspension, the wheel diameter had to comfortably exceed 36 mm.

In previous competition iterations, 65 mm wheels were utilized. However, they proved excessively large, creating severe vertical congestion between the drive motor, chassis braces, and LiDAR platform. Consequently, a target wheel diameter of ≈ 50 mm (48–50 mm) was selected, offering adequate ground clearance for the 36 mm motor while maintaining a compact vertical profile.

Commercial Evaluation & Shore Hardness Selection

Commercially available 2-inch (48 mm) FingerTech Combat Robot wheels were initially evaluated due to their high-traction tread and 40A durometer rating.

The durometer rating measures an elastomer's resistance to indentation on the Shore A scale, where lower values (20A–30A) indicate soft, gel-like compounds, and higher values (70A–90A) represent rigid plastics. A hardness of **40A Shore Hardness** represents the optimal balance for autonomous racing, providing exceptional surface traction and grip on the track mat without disintegrating or tearing under high motor acceleration torque.

Because 2-inch 40A combat wheels were unavailable domestically in India and prohibitively expensive to import, a custom in-house casting process was developed instead.

Manufacturing & Polyurethane Casting Procedure

To fabricate custom 48 mm high-traction tires, a two-part casting process was established using SLA resin 3D-printed molds and PLA wheel hubs. The Polyurethane selected was a 2 part 10:4 ratio 40A mixture made by Aditya Silicone. Standard teflon based mold release spray was used. *Please ensure proper PPE and safety when working with these materials.*

For the core wheel hub fabrication (`Wheel Hub.step`, see Figure 51), the part is 3D printed in PLA with an 18 mm width. To ensure a permanent mechanical bond between the printed hub and the cast tire, the outer cylindrical contact surface is printed with a "Fuzzy Skin" texture (0.1 mm thickness, 0.2 mm density), creating microscopic interlocks for the curing compound.

For the precision mold fabrication (`Tools/Tire Mold.step`, see Figure 54), a two-part split mold was 3D printed using Stereolithography (SLA) resin. SLA resin was selected over FDM because its smooth surface finish eliminates layer lines, allowing clean demolding without requiring draft angles. The mold halves clamp together using four M3 $\times$ 30 mm screws, washers, and M3 hex nuts.

During the polyurethane casting phase, a two-part liquid Polyurethane elastomer formulated to 40A Shore hardness is mixed, degassed to remove air bubbles, and poured through the mold's entry port until liquid emerges from the riser exit port. After a 24-hour cure cycle, the mold is unbolted, producing a balanced, high-traction polyurethane tire bonded securely to the PLA wheel hub.

A file called `Castings.csv` contains all the details of each and every individual castings, including Part A and Part B ratios, weights and remarks. It can be found in the `Other` folder.

2 PCB Assembly

The first question one should ask is why use a PCB at all? Why not just use jumper wires and other modules?

The answer is quite simple yet complex. A Printed Circuit Board (PCB) integrates all the power management components needed into a sleek and compact form factor without a lot of wiring. The Raspberry pi has a 20×2 pin interface, which isn't very neat in it self. One can connect the individual jumpers to the header pins but it becomes VERY cumbersome very fast. Hence a PCB is almost necessary for a vehicle that can be easily serviced and is well engineered. Another major driver for this, as you will learn, comes from issues learnt from Siddesh's participation in WRO the previous year. KiCAD was used as the eCAD tool to design the PCBs.

2.1 Purpose

When participating in WRO 2025, the SD Card we were using failed on us the day before the final competition. It was corrupted and the Raspberry Pi would not boot up at all. Turns out the issue was that that the Raspberry Pi was constantly undervolting under load. We had attempted to add a capacitor before the competition to resolve the issue, but the effort proved futile. Upon reviewing the logs, we confirmed that these undervoltage events were, in fact, what caused the SD card to fail. The buck converter was rated for 5V at a nominal current of 3A and a burst current of 5A. Yet the Raspberry Pi wouldn't boot.

The Raspberry Pi needs a voltage higher that just 5V. It needs a voltage of about 5.15V to prevent the undervolting issue we had seen earlier. We searched the market for a buck converter capable of providing the 5.1V and 3A but I couldn't find a suitable option. Hence it was determined that a custom solution was needed.

Hence the following considerations for the PCB were formed $\rightarrow$

1. Deliver 5.15V to the Raspberry Pi and have a continuous rating of atleast 3-5A to power the LiDAR as well.
2. Deliver 8V to the servo motor at 2A.
3. Provide a better hardware interface than the 20×2 connector for the Servo and ESC.
4. Be compact and reliable.
5. Have additional expansion ports for Time Of Flight Sensors, and more Servos.
6. Integrate the power switch and start switch, along with indicator LEDs needed for the rules on to the board.
7. Look good!

2.2 Form Factor

The first thing to decide after deciding the purpose is to decide the form factor of the PCB. This in turn, will decide where the PCB will live.

The best form factor is having the PCB as a hat above the Raspberry Pi itself. This is because of the following reasons →

1. There is a relatively large space above the Raspberry Pi, somewhere about 65 mm × 56 mm.
2. A simple header pin board can be added to the PCB, meaning it will directly interface with the header pins on the Raspberry Pi. This would eliminate the need for adding wires between the Pi and the PCB, ensuring efficient and smooth power distribution and connections.
3. It would provide ease of maintenance to the PCB and uninterrupted access to the switches.

2.3 Selection of the ICs

Due to the custom need for the 5.15V and a lack of existing modules, a custom voltage converter module would have to be implemented on the PCB hat. Since we were already using SMD (Surface Mount Device) components for the 5.15V PCB, we decided to integrate the 8V part as well on the PCB itself rather than have a separate module.

We use buck converters, which are a form of switching regulators rather than using linear regulators. Although linear regulators provide the smoothest possible voltage, they dissipate the remaining energy as heat, and hence are very inefficient and only good for low power applications where the voltage difference between the input and output is very low. However, we need the exact opposite. We need to draw an insignificant current and our voltage difference between the input and output is substantial. Hence we use switching regulators. The only downside is that they do not have a clean voltage output as they use a MOSFET to switch at between 0V and battery voltage at high frequencies. This can be compensated for using good design and with adding filtering capacitors at the output.

2.3.1 The 5.1V Rail

The LM61460 from Texas Instruments was chosen as the switching regulator due to its size and the fact that it has a continuous limit of 6A. It is also ideal because it has symmetric switching nodes and in built MOSFETs, which reduce noise.

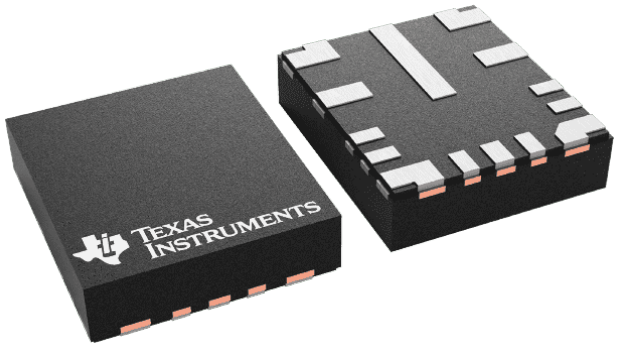


Figure 14: The LM61460 in the VQFN-HR package.

The values for the schematic were obtained using the TI Workbench.

2.3.2 The 8V Rail

The TPS62932 from Texas Instruments was chosen as it is incredible small and has a current rating of 2A. It also has an in built MOSFET in its incredible tiny package of just 2.1 mm × 1.6 mm.

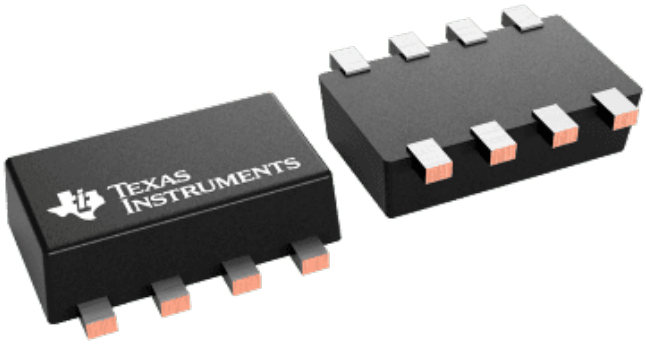


Figure 15: The TPS62932 in the SOT-583 package.

The values for the schematic were obtained using the TI Workbench.

2.4 The Schematic

The first stage of the PCB development process is creating the schematic; the wiring diagram for all the connections. The first step is to add all the individual components that are used, the Raspberry Pi, the buck converter ICs, the motors, the LEDs, the switches, the power connectors and any other I/O that maybe needed.

The switching regulators need additional hardware, such as resistors and capacitors along with an inductor. The value of these components decide what voltage you get at the output. To find these values, the TI workbench tool was used. The TI workbench allows one to configure the parameters and then produces a schematic (Figure 16). This is then used to set the value and the connections of the SMD components.

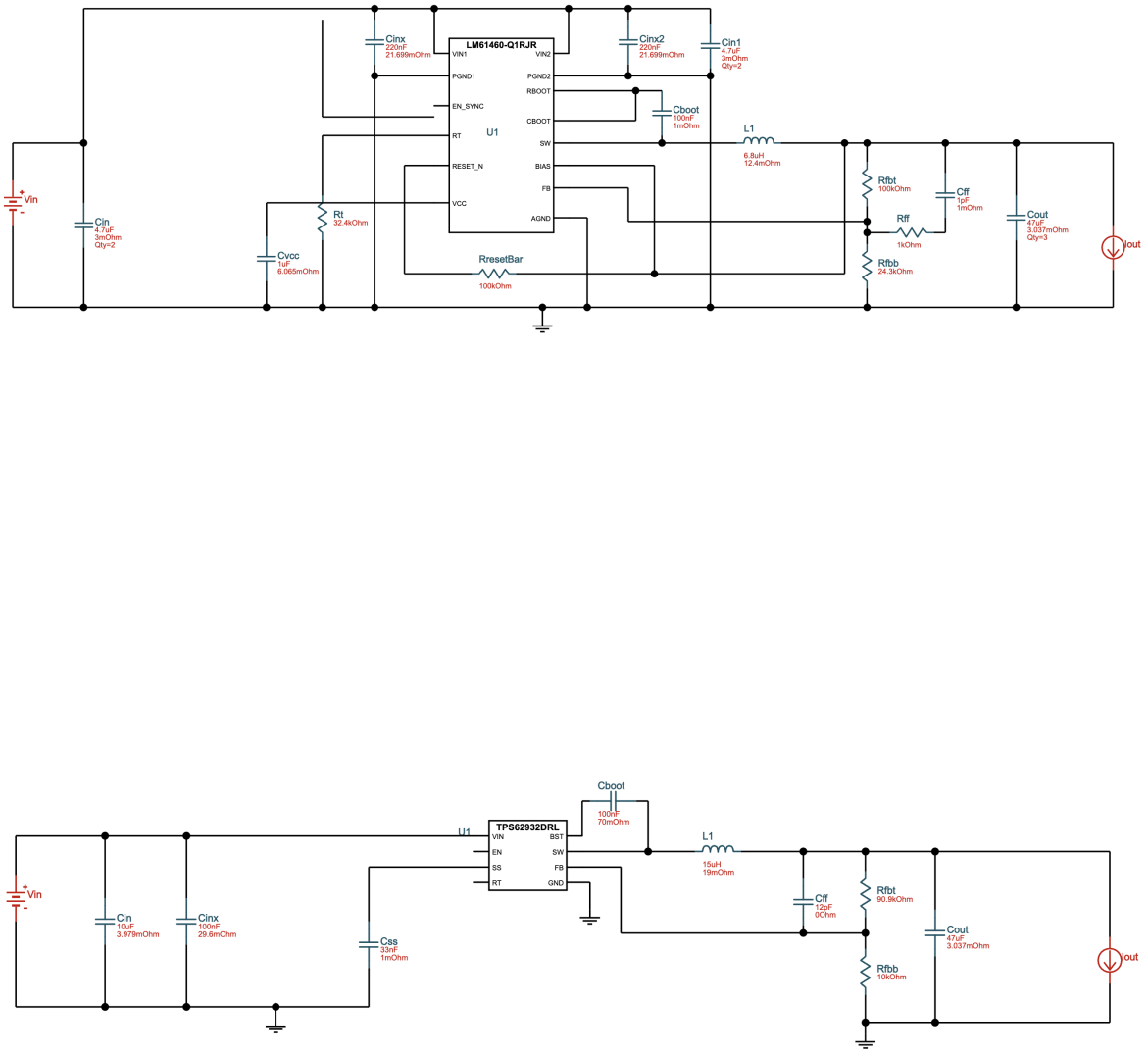


Figure 16: The files for the schematic given by TI Workbench.

The schematic (Figure 17) created is in 3 pages. The main page has the XT30 power connector input and the power switch. The Raspberry Pi page consists of the Raspberry Pi, the sensors and I/O devices. The power page consists of all the power regulation circuitry. It also has 2 additional electrolytic capacitors, one 220μF and the other 270μF. They both act as decoupling capacitors for the Raspberry Pi and the servo respectively. They ensure that when there are load spikes from either devices, the Raspberry Pi booting or the servo starting

to move, the spike does not reach the ICs and is handled locally.

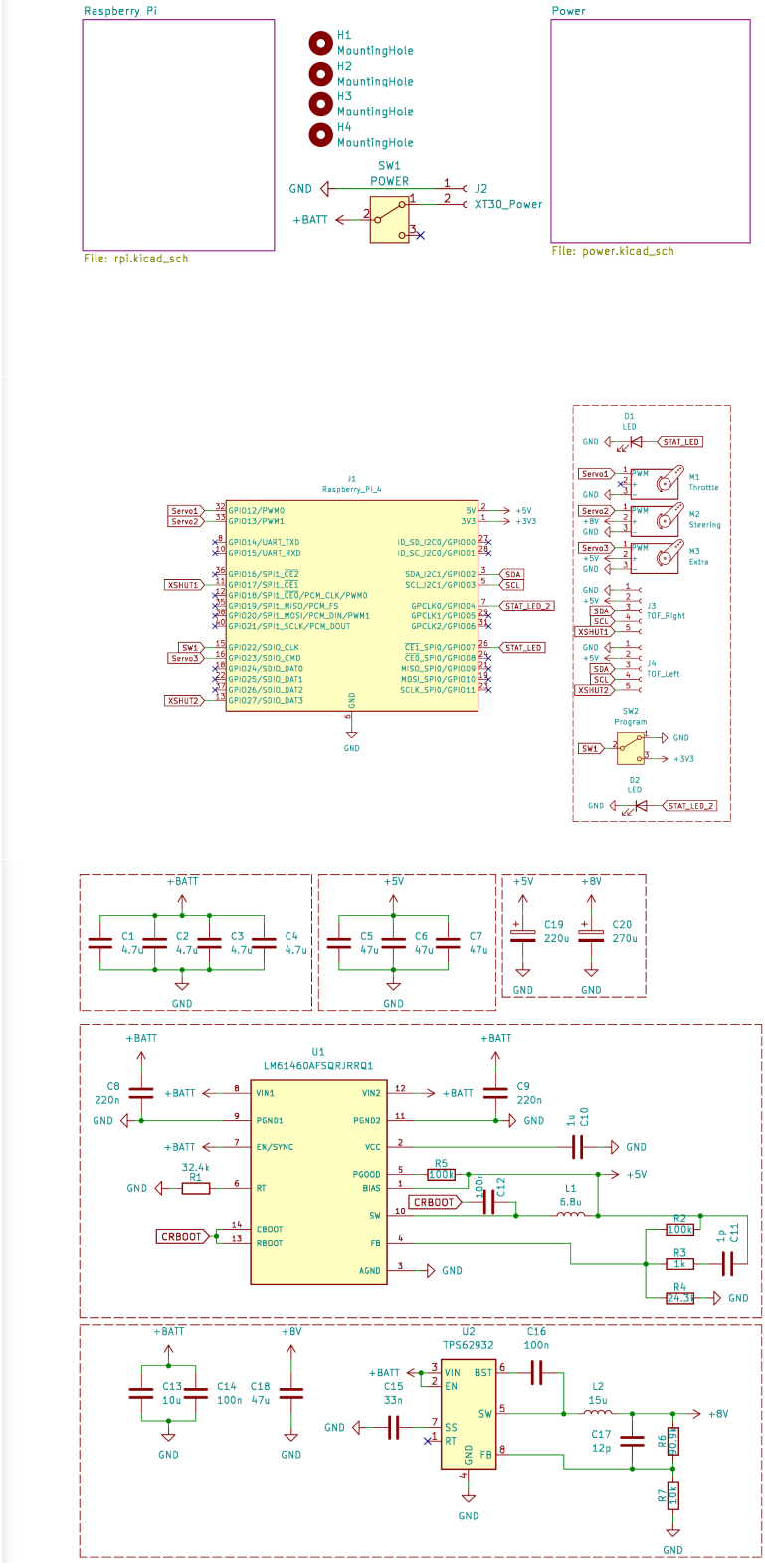


Figure 17: The 3 pages of the schematic.

The PDF for the Schematic can also be found in the PCBA folder. The BOM for all the parts is also available there, along with the manufacturer used.

2.5 The Layout & Routing

This step is very crucial, it is responsible for how well the components interact with each-other and determines how clean your output voltage is. A **4 layer board** was chosen. This was done as it would make the routing easier across multiple layers and because a 4 layer stack up is recommended in the LM61460 data sheet.

All the SMD components would be place on one side of the board so that the board could be assembled on a hot plate. This would also aid with the ease of assembly. The following is the stack up →

1. Layer 1: It carries much of the important **signals**, especially for the buck converters.
2. Layer 2: A solid **Ground** plane to reduce EMI and act as a fast return path of the top player.
3. Layer 3: The **Power** plane consisting of the input voltage, 5.15V and 8V.
4. Layer 4: A **Ground** plane that also carried some of the **signals** from the Raspberry Pi.

The first thing to do is create the board outline. A rounded rectangle with length 65mm, height 56mm and a corner radius of 2mm was chosen. A 6mm notch was then removed from the bottom right of the board to allow the camera ribbon cable to pass through. Figure 18 shows the shape of the board.

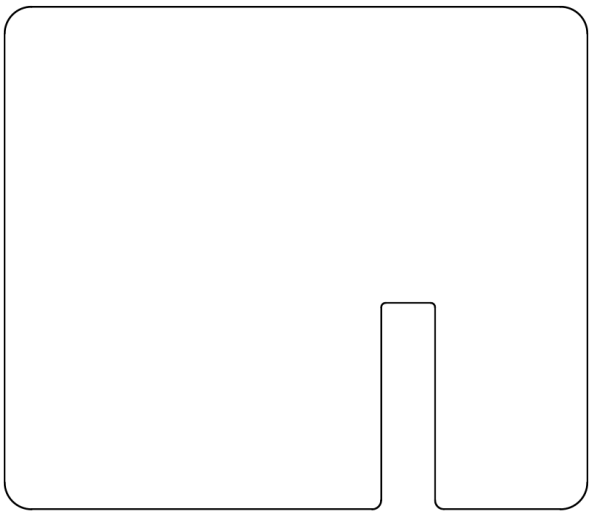


Figure 18: The "edge cuts" layer.

After this, the fixed components are place, such as the mounting holes for the board and the 20×2 female header pin that connects to the Raspberry Pi. These need to be incredibly precise as they are the locating hardware for the hat to connect on to the Raspberry Pi. M2.5 holes are used.

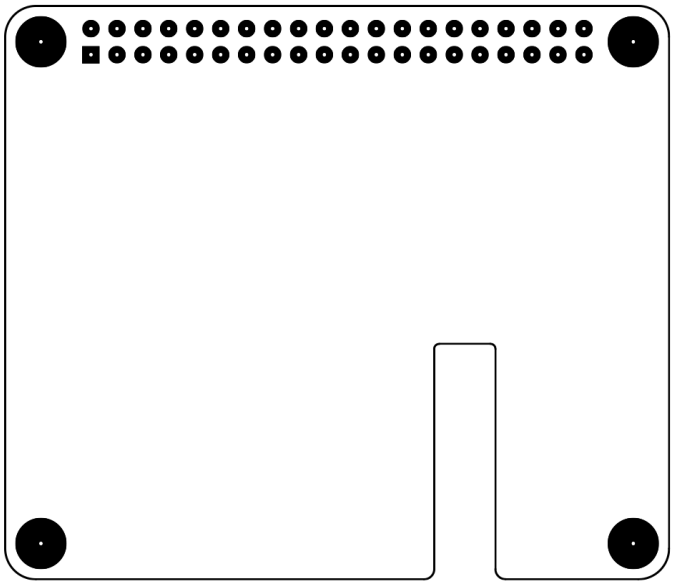


Figure 19: The fixed items layer.

The next part is the most cumbersome. All the physical parts need to be laid out in a manner that ensures the most ideal connections possible. The 3 main current loops; ON-State Current Loop, OFF-State Current Loop and The "Hot Loop"; of the buck converter need to be kept as short as possible. Since we are using a symmetric buck converter, it also needs to be made sure that the power lines to both sides are as similar as they can be. The path between the switching node and the inductor needs to be as short as possible. All the connectors need to accessible.

For this reason, the power connector (XT30) and the power switch were placed on the right side. The switching circuitry was spread around the middle. The second switch and the connectors for the motor and servo were kept on the left side. The electrolytic capacitors were placed as close to the outputs as possible. The LEDs were also placed in the middle. The extra components, such as the connectors for the 2 ToF sensors and an extra PWM connector were kept on the bottom and bottom right respectively. Figure 20 shows the layout of the final board. It is fairly crammed but a lot more items, such as more LEDs, Current and Voltage sensing and a CAN BUS port can also be added in the future.

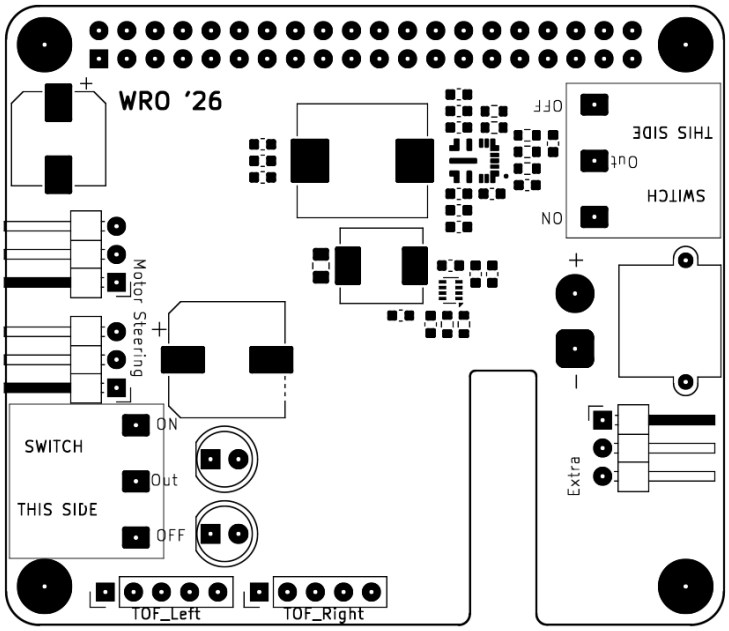
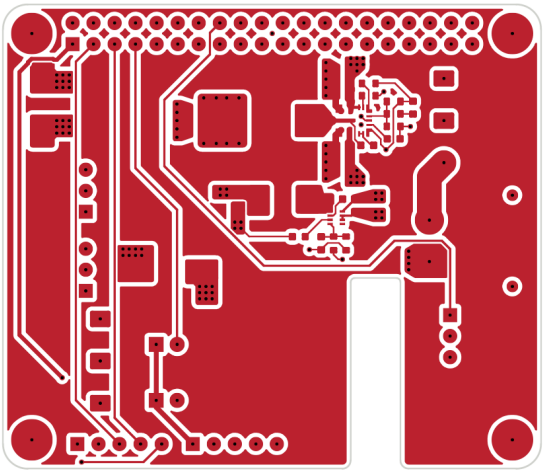


Figure 20: The final layout depicting the placement of all the components.

Figure 21 shows the actual copper layers as they are. Copper pours are preferred whenever possible. They also help with thermal management. It was made sure that there were no wires beneath the two inductors to make sure all the signal traces are as clean as possible.



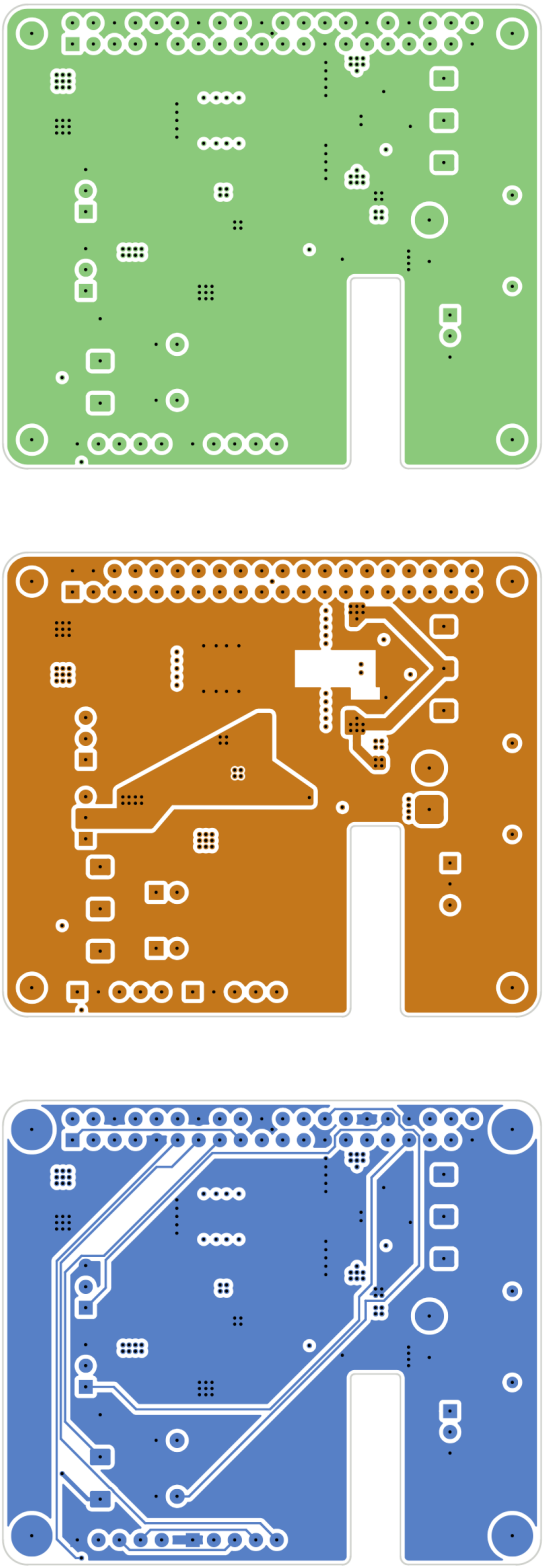


Figure 21: The 4 copper layers. Layer1 → 4.

Some art work and quotes were added to the free space to ensure complete utilization and occupation of left over space. A black solder mask was chosen. Figure 22 show the PCB rendered in the KiCAD 3D viewer. The contribution for the art work goes to M.C. Escher.

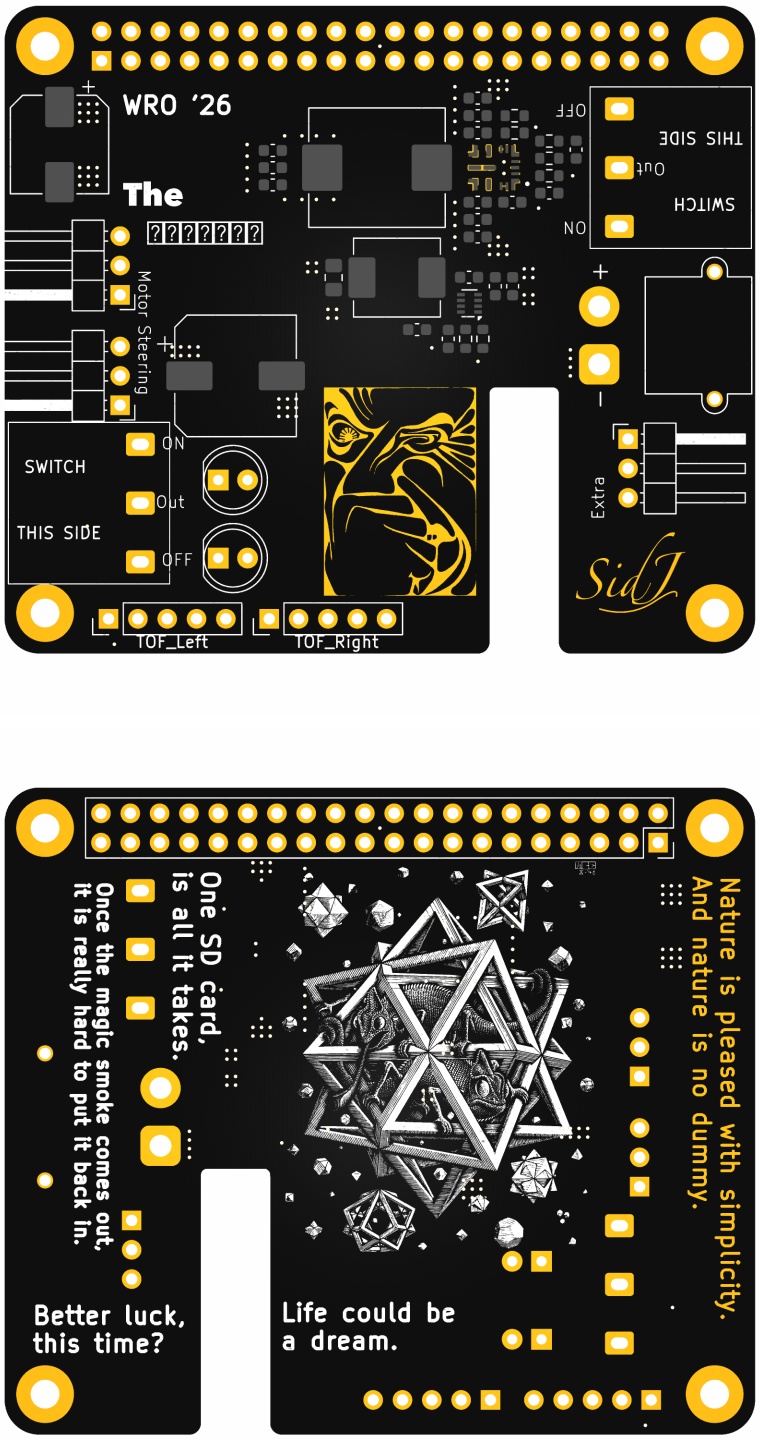


Figure 22: The rendered PCB.

The PCB was manufactured by JLC. JLC was chosen due to their lower cost for black solder mask compared to Indian manufactures. The following are the basic parameters:

1. Base material: FR4
2. PCB Thickness: 1.6mm

3. PCB Color: Black
4. Silkscreen: White
5. Material Type: FR4 TG135
6. Surface Finish: Lead Free HASL (Hot air solder leveling)

2.6 Assembly

The assembly was done by hand. 0603 class components are not too hard to place by hand. Alternatively, assembly can be done by the manufacturer as well.

A stencil was used to apply the layer of 183° C solder paste on to the board with the help of an old credit card. The "Interactive Html Bom" plugin (Figure 24) was used to aid with assembly. The plugin creates a interactive html file that can be used to figure out the placement of the components. Alternatively, the PCB and schematic editor can itself be used to figure the placement out but using the plugin is recommended do to its convenience. Once the components have been laid out, a final visual inspection can be carried out.

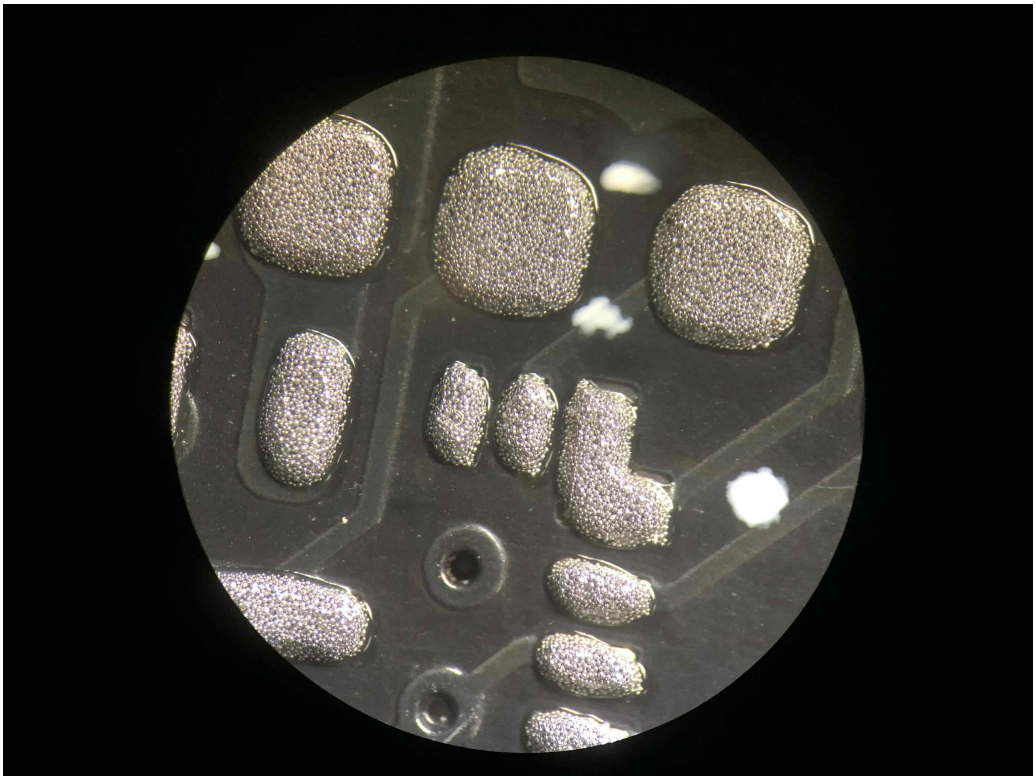
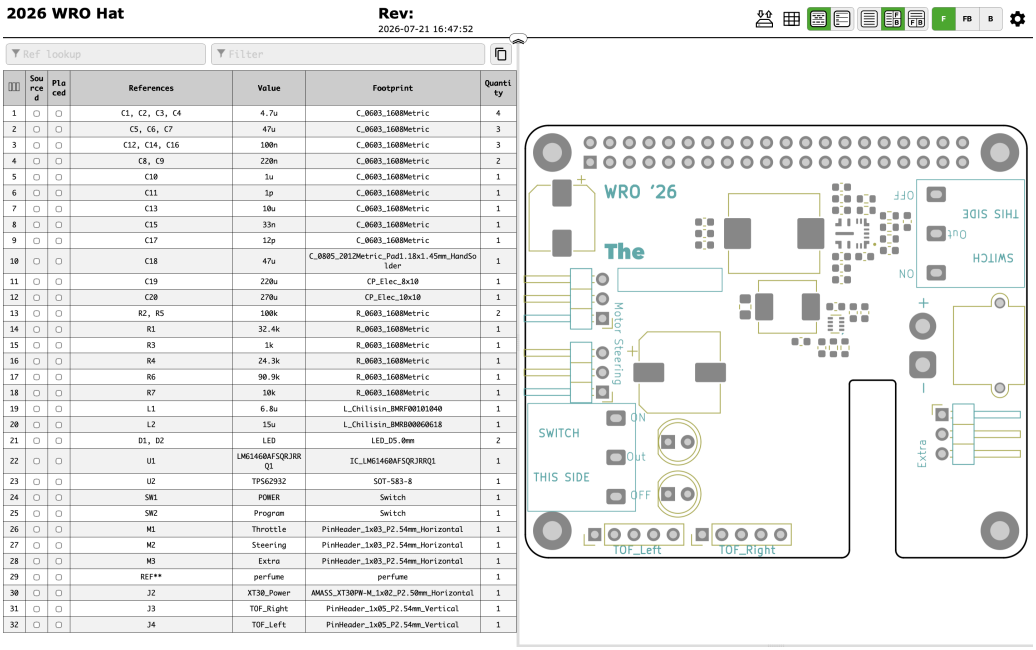


Figure 23: The solder paste on the pads after being applied with a stencil.

The boards can then be re-flowed on the hot plate. It is recommended to follow the heat curve of the solder paste being used. If the plate available does not have the ability to follow a curve, let the board heat soak at about 50-80° C below the melting point of the solder paste for about 1-2 minutes to let the flux activate before increasing the temperature to its final point.

Our hot plate did not have the feature to follow the heat curve so we let our boards soak at 130° C for 90 seconds after increasing the temperature to 195° C till all of the paste re-flowed and waited 30 seconds more before taking the boards off the hot plate.



	Sourced	Placed	References	Value	Footprint	Quantity
1	○	○	C1, C2, C3, C4	4.7u	C_0603_1608Metric	4
2	○	○	C5, C6, C7	47u	C_0603_1608Metric	3
3	○	○	C12, C14, C16	100n	C_0603_1608Metric	3
4	○	○	C8, C9	220n	C_0603_1608Metric	2
5	○	○	C10	1u	C_0603_1608Metric	1
6	○	○	C11	1p	C_0603_1608Metric	1
7	○	○	C13	10u	C_0603_1608Metric	1
8	○	○	C15	33n	C_0603_1608Metric	1
9	○	○	C17	12p	C_0603_1608Metric	1
10	○	○	C18	47u	C_0805_2012Metric_Pad1.18x1.45mm_HandSolder	1
11	○	○	C19	220u	CP_Elec_8x10	1
12	○	○	C20	270u	CP_Elec_10x10	1
13	○	○	R2, R5	100k	R_0603_1608Metric	2
14	○	○	R1	32.4k	R_0603_1608Metric	1
15	○	○	R3	1k	R_0603_1608Metric	1
16	○	○	R4	24.3k	R_0603_1608Metric	1
17	○	○	R6	90.9k	R_0603_1608Metric	1
18	○	○	R7	10k	R_0603_1608Metric	1
19	○	○	L1	6.8u	L_Chilisin_BMRF00101040	1
20	○	○	L2	15u	L_Chilisin_BMRB00060G18	1
21	○	○	D1, D2	LED	LED_D5.0mm	2
22	○	○	U1	LMG1460AFSQRJRRQ1	IC_LMG1460AFSQRJRRQ1	1
23	○	○	U2	TPS62932	SOT-583-8	1
24	○	○	SW1	POWER	Switch	1
25	○	○	SW2	Program	Switch	1
26	○	○	M1	Throttle	PinHeader_1x03_P2.54mm_Horizontal	1
27	○	○	M2	Steering	PinHeader_1x03_P2.54mm_Horizontal	1
28	○	○	M3	Extra	PinHeader_1x03_P2.54mm_Horizontal	1
29	○	○	REF**	perfume	perfume	1
30	○	○	J2	XT30_Power	AMASS_XT30PW-M_1x02_P2.50mm_Horizontal	1
31	○	○	J3	TOF_Right	PinHeader_1x05_P2.54mm_Vertical	1
32	○	○	J4	TOF_Left	PinHeader_1x05_P2.54mm_Vertical	1

Figure 24: The Interactive HTML BOM Tool.

The HTML file generated by the plugin can be found in the PCBA folder along with the BOM.

Once this is done, the through hole components; like the switches, the power connector and the 20×2 header connector; can be soldered on to get the final board.

Note: Please refer to the next subsection, Mistakes & Errors before powering on the board. There is a 1000 μF capacitor to be added between the output of the switch and GND.

2.6.1 Advice

The following is the recommended order for the placement of the components ⇒

1. U1
2. U2 (be careful, some wiggling around is needed)
3. C8, C9
4. C1, C2, C3, C4
5. C5, C6, C7
6. C18
7. R1
8. R2, R5

9. R3
10. R4
11. C10
12. C11
13. C13
14. C12, C14, C16
15. C15
16. C17
17. R7
18. R6
19. L2
20. L1
21. C19
22. C20

2.7 Mistakes & Errors

The first iteration, **V1.0**, had a fatal flaw. During the schematic design, the enable pin of the LM61460 was connected to ground. The enable pin needs a voltage of greater than about 3V. This mean that the chip would be in the "shutdown mode". This mean it would not work at all.

To attempt fix this issue, the trace from the GND via to the EN pin was cut off using a razor blade. A thin copper wire (32 AWG) was then added very carefully between the VIN and the EN pad. Typically, this is called a "**bodge**" (Figure 25). This turned out to be very unsuccessful. This was because of the Package of the IC and the microscopic, but still substantial thickness of the wire. The VQFN package has flat pins at the bottom that interface with the surface directly (Figure 14). This meant that while the wire was fairly thin, its thickness meant that not all the pins of the IC made contact with the PCB. It was sort of like a see saw. Another attempt was made with a thinner strand of wire, about a third of the diameter but that was also unsuccessful.

Hence the decision was made to re order a new set with the fixed connections, **V1.1**. The original error was traced down to a mistake in copying the schematic from TI workbench to the KiCAD schematic editor.

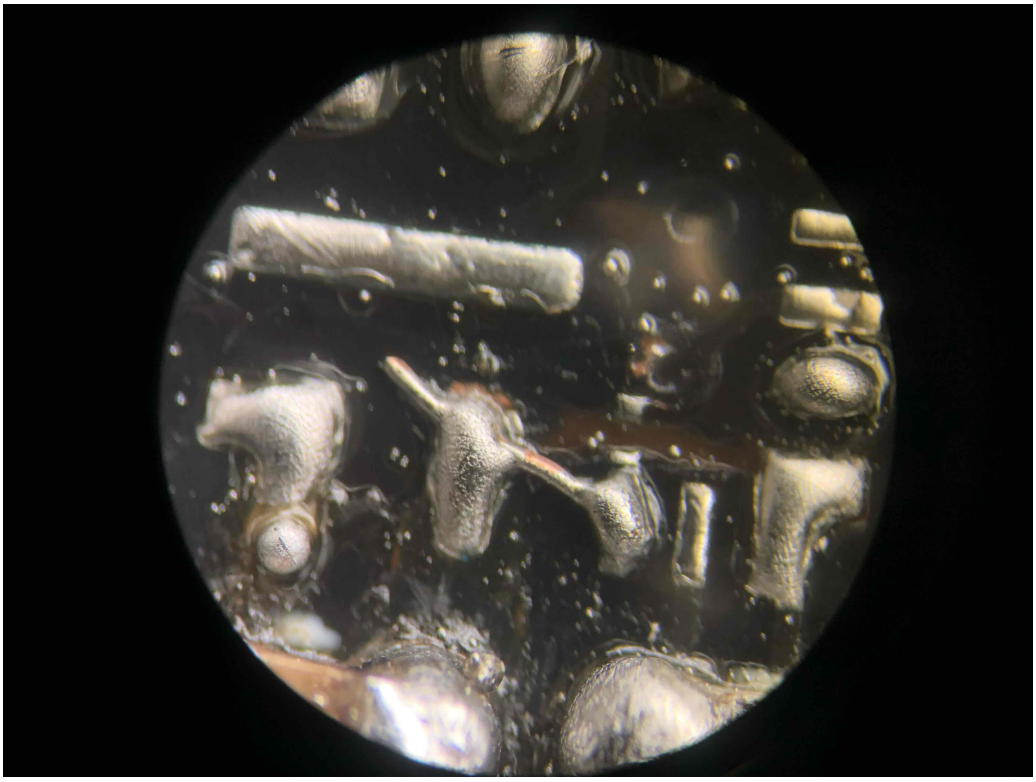


Figure 25: The brown area on the top right of the copper wire represents the cut out trace.

The newer version, **V1.1** was assembled successfully and worked flawlessly, providing 5.15V to the Raspberry Pi as intended. During hardware testing, when using the switch to power on and off the Raspberry Pi, the LM61460 chip, literally, burst into flames and released the magic smoke. This was right as the switch had been toggled to the on position, triggering a catastrophic hardware phenomenon known as an LC hot-plug transient spike. The wires from the battery inherently act as an inductor, and the ultra-low ESR (Equivalent Series Resistance) ceramic input capacitors on the board act as a capacitor, together forming an undamped LC resonant circuit. When the mechanical switch was flipped, its internal contacts bounced microscopically. This rapid connect-disconnect action, combined with the near-zero resistance of the ceramic capacitors, caused the sudden inrush current to ring like a bell. The voltage overshot significantly, effectively doubling the 16.8V battery input and easily punching through the 36V absolute maximum rating of the LM61460.

This destructive spike was further amplified by the DC bias derating of the ceramic capacitors, which temporarily lose capacitance under high voltage, forcing the transient spike even higher. The fix for this required introducing a "shock absorber" to dampen the resonant circuit. By adding a 1000 μF electrolytic capacitor directly between the output of the switch and ground, the naturally higher ESR of the electrolytic component safely dissipated the kinetic energy of the inrush current as microscopic heat. This completely arrested the pendulum swing of the LC ringing, clamping the voltage safely to the battery level and bulletproofing the circuit against future switch bouncing.

NOTE: The PCB design files do not include this fix, and it is to be implemented after.

3 3D Prints & CAD

The body of the vehicle is primarily made up of 3D printed parts. Onshape was the CAD package used. Extreme considerations were made when designing the vehicle. Ease of reputability, along with structural rigidity were the utmost priority when designing the parts. This was exasperate from the constraints already present when using FDM (Fused Deposition Modeling) 3D printing.

Tolerances for the parts were borrowed where ever possible as a foundational design principle to ensure predictable and consistent assembly. Because FDM printing is inherently prone to dimensional variations; caused by thermal shrinkage, layer lines, and extrusion inconsistencies; calculating unique clearances for every single joint is highly inefficient and risks assembly failure.

By standardizing and "borrowing" proven tolerance profiles (such as a baseline 0.2mm gap for slip fits and 0.4mm for clearance holes) across the entire CAD model, the design creates a modular, standardized system. This strategic allocation allows non-critical dimensions to absorb the printer's inherent inaccuracies. Ultimately, this prevents tolerance stack-up issues and guarantees that structural interfaces bolt or snap together seamlessly on the first print, eliminating the need for tedious manual post-processing, filing, or reprint iterations.

Pre-flight and Orca Slicer were the Slicers used for the conversion of the 3D files into G-Code. The utmost care was given to the parts in the design process. There are no large parts that need a massive printer to be printed. All the parts can fit on a minimum build volume of about $150 \times 150 \times 150$ mm. The whole assembly was created in Onshape to ensure parts fit together seamlessly.

3.1 Layout & Wheelbase

To adhere to the strict volume limitations we set in the introduction, a slightly wider wheelbase was opted compared to what most competitors would probably use. This was done so that more horizontal volume could be allotted to the components without shifting them vertically. A strict horizontal width limit was put at 180mm to ensure the 200mm limit was not breached. The wheels where placed as far out as possible to maximize component volume and have better stability.

A wheelbase of 184.3mm was chosen. A longer wheelbase would mean that the vehicle would have trouble parallel parking. The number 184.3, may seem odd but it was derived from scaling the Aston Martin Valkyrie down to the size of the wheels being used.

The layout, as mentioned earlier, a Rear Wheel drive mechanism with front wheel steering was chosen. This made the layout slightly easier but it was crammed nevertheless. Figure 26 shows the CAD assembly, moving top down.

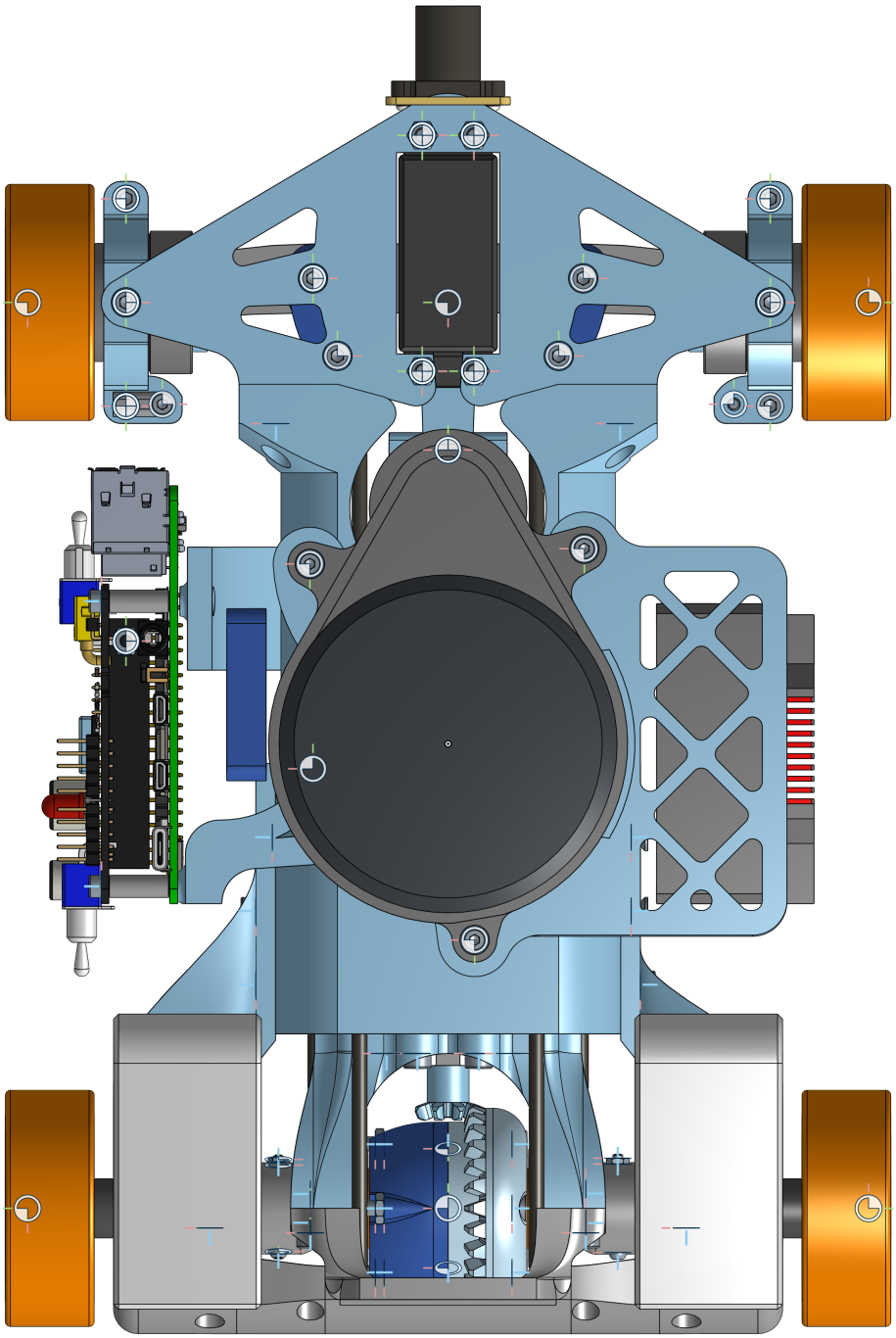


Figure 26: The Onshape assembly.

Starting at the front, we have the camera as the front most element. This is driven by the fact that the camera has a very high FOV and the best view would be unobstructed. Just behind it is the front axle. It houses the steering system and the bearings. In the center there sits the LiDAR, directly positioned on top of all the components. Below the LiDAR, with a

healthy amount of spacing to prevent EMI is the motor. To the right of the motor, the ESC is mounted vertically. The Raspberry Pi is mounted on the left side of the motor. Moving further back, we reach the differential and the rear axle. The batteries sit atop the rear axle baring mounts.

It was critical for every single part to be accounted for. All the models were self designed. There are a few critical assemblies that need to be discussed in order.

3.2 The Steering Assembly

Front Wheel Adapter

The steering assembly originates from two points in space, the contact points for the needle roller thrust bearings (Figure 9). The bearings connect to the wheels via a rod that is capped at on end. The rod as a nominal diameter of 12mm and a length of 38mm. The cap at the end has a diameter of 20mm and a width of 3mm. The rod also has a ⌀2mm hole at the end, spaced 4.5mm from the edge of the non capped end to interface with the wheel (Figure: **??**) via a M2 screw.

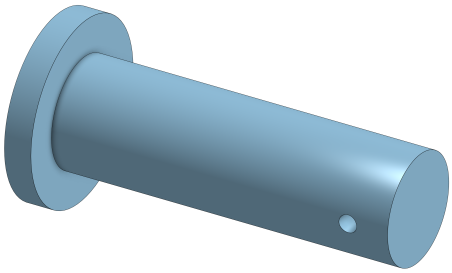


Figure 27: The Front Wheel Adapter Pin.

The pin is printed vertically, with the cap end facing down. This does spread the load across the layer lines, the load on the front wheels isn't drastic enough for one to be concerned. The part is printed with a layer height of 0.2mm, has 3 perimeters and a 25% infill.

Steering Knuckles

We next interface with the stationary parts of the bearing. To do this, we make a clamp style part that relies on the pressure and friction to keep the bearing constrained. We replicate the grove in the bearing as a positive feature to constrain the bearing axially.

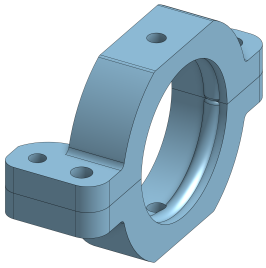


Figure 28: The Steering Knuckle.

To interface the two part clamp, we use two M3 $\times$ 12 screws and M3 nuts. We flatten out the sphere at ⌀31mm so that the clamps can interface with the other steering parts and rotate in the vertical axis. To do this, we add a ⌀2.8mm hole on the top for a M3 screw. We keep the hole smaller than the screw diameter so that the screw can create the threads in the plastic part as it enters. This is done as using a heat set inset would take too much space. Also it is justified by the lower loads. We also add a ⌀2.5mm hole so that a screw can be added for the steering rod interface. These would be like Steering Knuckles in a traditional car. It is mirrored to fit on the left side.

The part is printed with the face closest to the bearing (and image in Figure 28) facing down. This not only transfers the loads along the layers but also ensure a good interference fit with the bearing. The part is printed with a layer height of 0.2mm, has 3 perimeters and a 25% infill.

Steering Column

The steering column consists of 4 parts (See Figure 29), the top brace (light blue), the bottom brace (grey) and 2 side braces (dark blue). It mounts the servo on the center of the front axle and provides interface points with the rest of the vehicle.

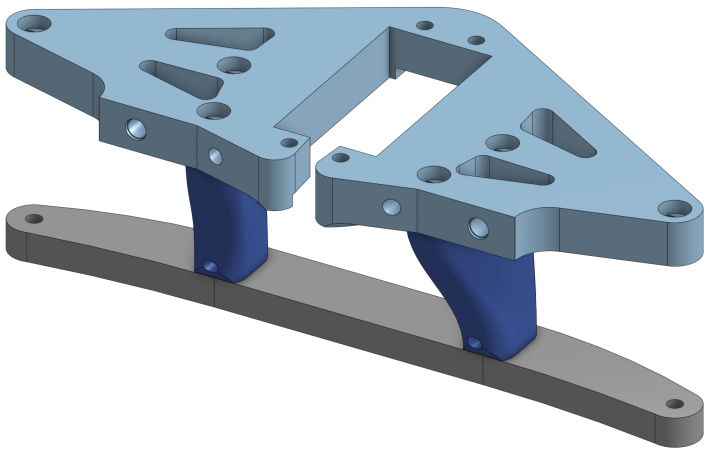


Figure 29: The Steering Column.

The top brace is the largest part in the steering column. It mounts the servo and the camera. The servo mounts via 4 M3 $\times$ 20 screws and has a mounting pattern of 10.5 $\times$ 48mm. The top

brace has an hole at each extreme so that it can interface with the steering knuckle. It does so by using M3 $\times$ 8 screws. The holes are space 131mm apart. The top brace has 4 additional counter bore holes that connect it with the 2 side braces. On the rear face of the mount, there are two locating holes for the carbon fiber rods that are spread apart by 36mm. There also exists a flat face with space for 2 $M3 \times 6$ heat set inserts so that the part can interface with the rest of the body and is constrained.

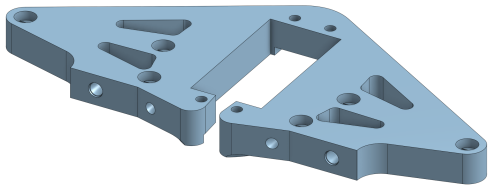


Figure 30: The Top Brace.

The part is printed with the top face (See Figure 30) facing down, at a layer height of 0.2mm, has 3 perimeters and a 25% infill. The part nominally includes features to aid with counter bore bridging but PreFlight can also be used to aid with this.

The bottom brace is the simplest part. It is a slightly curved piece that spans the breadth of the wheelbase. It has 4 countersunk holes, 2 to interface with the steering knuckles, and 2 to interface with the side braces, using M3 $\times$ 8mm countersunk and M3 $\times$ 10mm countersunk screws respectively. The part is printed with the bottom face (See Figure 31) facing up, so that the sunk bores are not printed as overhangs. It is printed at a layer height of 0.2mm, has 3 perimeters and a 25% infill.

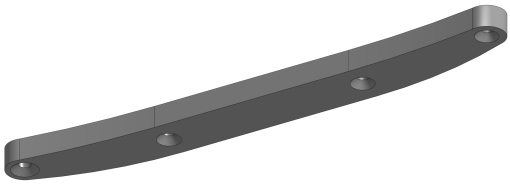


Figure 31: The Bottom Brace.

The side braces are slightly more complex in geometry as they are a product of lofts. The side braces have spaces for 2 M3 $\times$ 6mm brass heat set inserts on the top, and 1 M3 $\times$ 6mm brass heat set inset on the bottom to interface with the top and bottom braces receptively. The side brace also has a feature to interface with the bottom set of ⌀3mm Carbon Fiber Rods. The part is designed for the left and is mirrored to fit for the right side. The part is printed with the top surface facing down, at a layer height of 0.2mm, has 3 perimeters and a 25% infill.

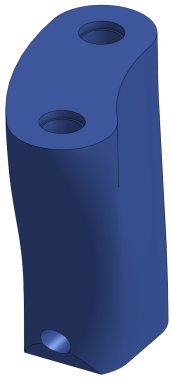


Figure 32: The Side Brace.

They all combine to form the steering column.

Servo Horn

The servo horn is the part that clamps on the the shaft of thew servo and interfaces with the steering linkage rods. It is a fairly simple part. It houses a ⌀6mm hole at one end and the space for a M3 × 6 spacer (See Figure 33, grey) that is press fit into it. The distance between the two is 30mm. The height of the part is also 6mm. The part is printed at a layer height of 0.2mm, has 3 perimeters and a 25% infill. To maximize the frictional force, the inside surface of the hole is printed with Fuzzy Skin, with a thickness of 0.1mm and 0.2mm.

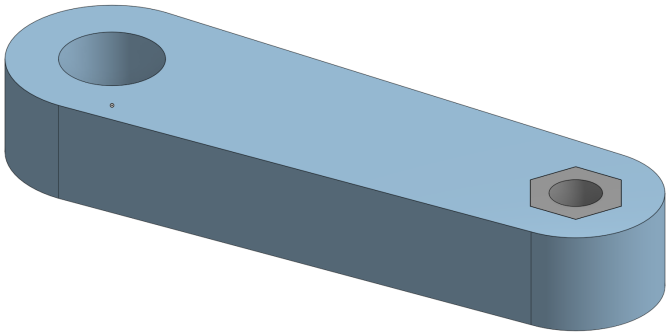
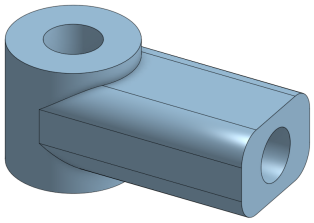


Figure 33: The Servo Horn.

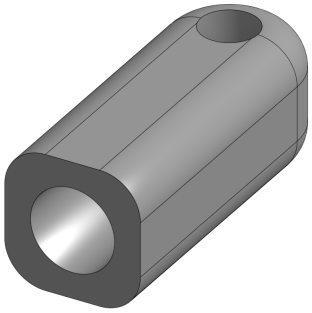
Steering Links

The steering links, along with a ⌀3mm Carbon Fiber rod make up the steering rod. The links have 2 parts (See Figure 34), one that connects to the servo, the servo link (Figure 34a) and the other other that connects to the knuckle (Figure 34b), the wheel link. The servo link has a height of 7.1mm with a ⌀3mm hole running through the center for a M3 × 10mm screw. It also has a space for the carbon fiber rod perpendicular to the screw. The wheel link is similar with a height of 5mm and a hole for a M2 × 20mm screw that connects it to the steering knuckles. The two links rely on the screw as a "pin" to rotate freely. A bearing is not used for this revolute

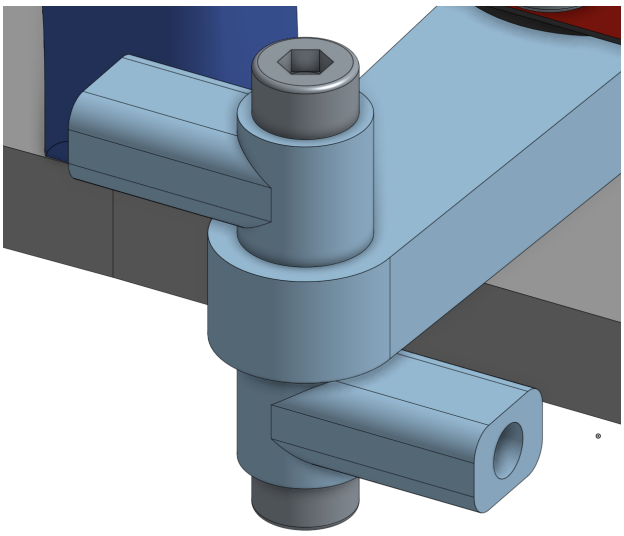
joint as the size is too small to accommodate any and the force are low. The part is printed at a layer height of 0.12mm, has 3 perimeters and a solid infill. 2 sets of these are needed to form the 2 steering rods. One is mounted at the top and the other at the bottom (Figure 34c).



(a) Servo Link



(b) Wheel Link



(c) Installation Context

Figure 34: The Steering Links

TPU Spacer/Shim

A Outer ⌀16mm and inner ⌀12.80mm spacer, with a height of 6mm is needed. This sits between the bearing and the wheel and provides pre-load to the whole stack up. This is used as the roller thrust bearing is only constrained in one direction by the end cap. The other end is hence constrained by the TPU spacer and the pre-load it provides. The pre-load also ensures there is as little play between the axle and the bearing as possible. Figure 35 (red) shows the placement of the spacer in section view. Since it is made out of a flexible material with a shore hardness of 95A, the height of the spacer determines the pre-load force. The part is printed at a layer height of 0.2mm, has 3 perimeters and a solid infill. It is utilized on both sides.

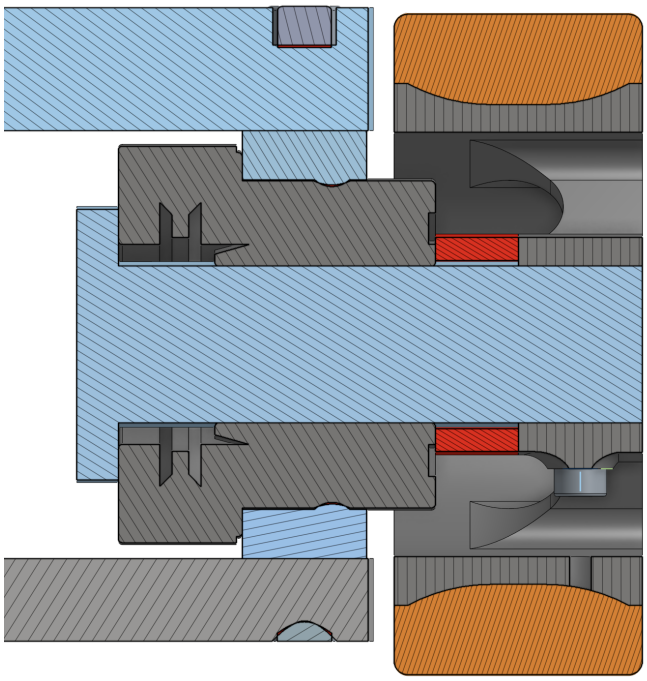


Figure 35: The TPU Spacer.

This concludes the assembly of the front axle. The length of the ⌀3mm carbon fiber rod used in the steering links is ≈ 48mm. It is glued to the steering links.

3.3 The Differential

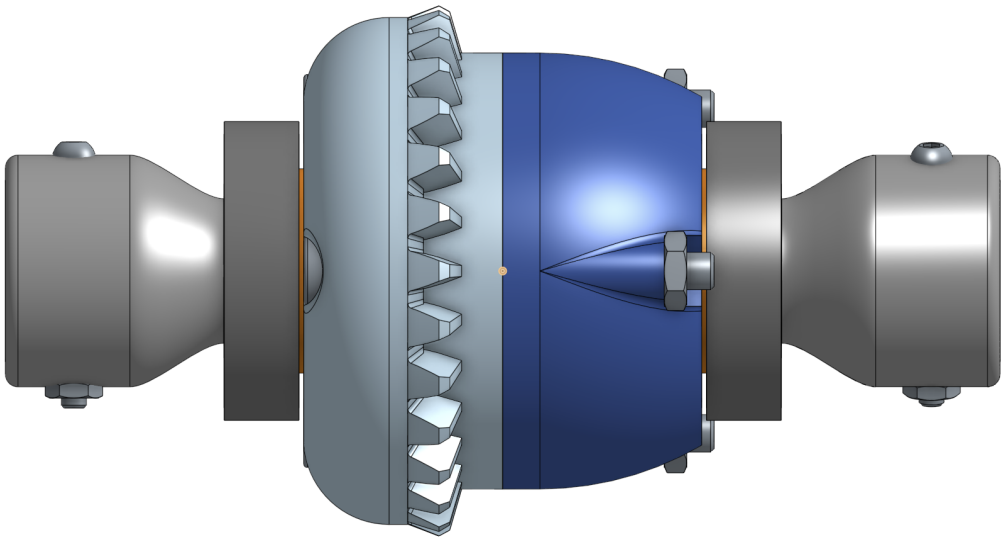


Figure 36: The Differential Gearbox.

As mentioned earlier, a differential gear box is used. The center of the gearbox is formed by the "spider" gears and the carrier along with the main gearbox. In a differential gearbox, the housing containing the spider and main gears also rotates and is mounted to a stationary

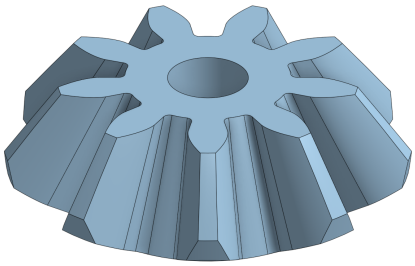
mount. The housing contains the main gear reduction for the differential. The spider gears assist with the difference in speeds of the two main gears (and hence wheels). All of the parts of the gearbox are 3D printed. The net reduction produced is $8 : 28 = 2 : 7$.

The Gears

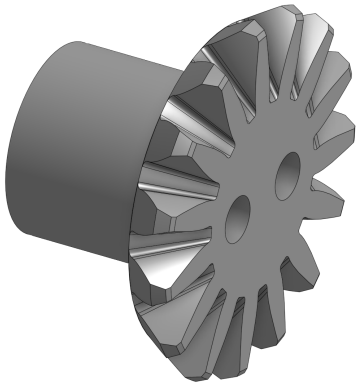
The differential consists of 3 spider gears and 2 main gears All the gears present are bevel gears created using the OnShape bevel gear scripts.

The spider gears have 9 teeth, a module of 1.5mm, a pressure angle of 20° and a bevel angle of 30.96°. The spider gears have a ⌀3mm hole through the center to interface with the axle. The top has be flattened to aid with printing. They are set 120° apart. See Figure 37a.

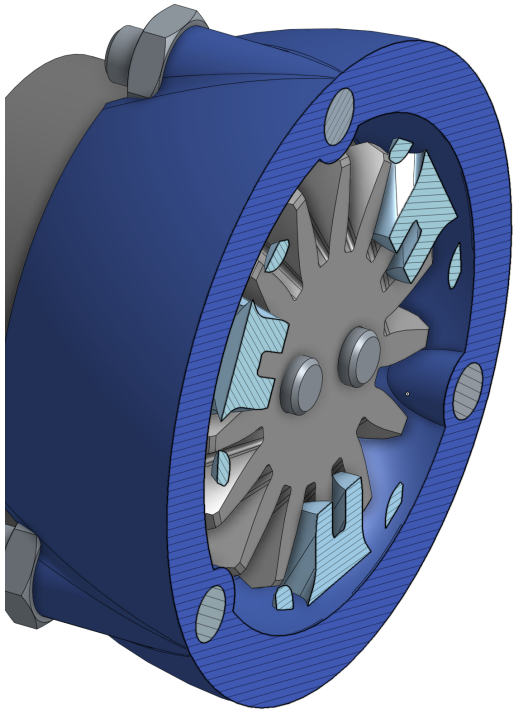
The main gears have 15 teeth, a module of 1.5mm, a pressure angle of 20° and a bevel angle of 59.04°. Notice that the bevel angles of the 2 types add up to 90°. The main gear transfer all the load to the wheels. Hence, using a single screw in the center would cause it to loosen due to the torque caused. Hence, 2 screws are use that are placed 3mm apart from the center to connect the gears to the part that interfaces with the rear axles. Behind the gear sits a ⌀12mm ×11mm cylinder that interfaces with the bearings. The top has be flattened to aid with printing. See Figure 37b.



(a) The Spider Gear



(b) The Main Gear



(c) Installation Context

Figure 37: The Differential Gears

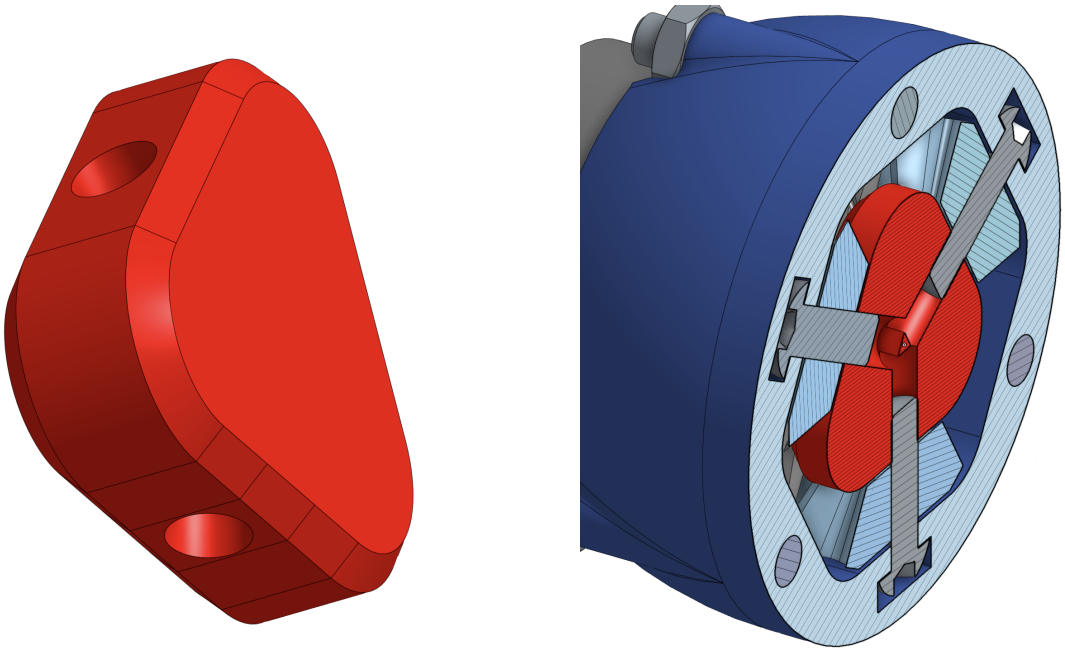
All 5 of the parts are printed at a layer height of 0.12mm, has 3 perimeters and a solid infill.

A brim is used to aid with bed adhesion. The spider gears and the main gears are printed with the gear face facing down. This means that the forces of the gears runs along the layer. However, this means that the rotational force of the main gear to the wheels runs across the layer lines, which would mean that the are where the gear ends and the cylinder starts is the weakest area. To counter this, the M3 × 20 button head screws that are used to connect the main gear and the axles run all the way to the end of the gear, meaning they transfer most of the rotational loads and strengthen the part's function across the layer lines. See figure 37c.

The Carrier/Spider Center

The Spider Center is the part that holds all the spider hears in the center and aligns them. In the housing, it is suspended by the M3 × 10mm Button Head screws. See Figure 38.

It's a filleted triangle with 3 ⌀3mm holes on each side. It has a height of 6mm. It is printed at a layer height of 0.12mm, has 3 perimeters and a solid infill. A brim is used to aid with bed adhesion. The part can be printed on either of the larger, flat surfaces.



(a) The Part. (b) The Spider Center in Action.

Figure 38: The Spider Center.

Housing

The housing for the differential gears consists of 2 halves, the left half and the right half. The two halves are connected by 3 M3 × 30mm screws and M3 nuts (See Figure 36).

The housing not only houses the gears but also 2× 6901ZZ bearings (Figure 10). This is to decouple the rotation of the main gears with the rotation of the housing when the two axles are spinning at different speeds. The left housing and the right housing are nearly identical. The left housing houses the bevel gear ring needed to rotate the differential housing using the motor. That gear has 28 teeth, a module of 1.5mm and a bevel angle of 70°. In Figure 39,

the left housing is the one in light blue and the right housing is in dark blue. The bearings sit inside the housing.

The part is printed with the center surface of each of the half facing down. Looking carefully, a chamfer is present on the end where the bearings sit. This allows for the overhangs to be managed better. It is printed at a layer height of 0.12mm, has 3 perimeters and a 25% infill. A brim is used to aid with bed adhesion.

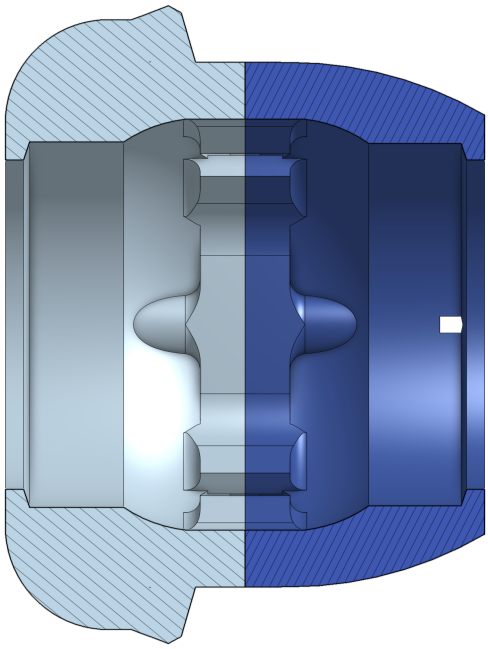


Figure 39: The Housing for the Gearbox.

The Attaching Cups

The Cup is the part that interfaces with main gear and the ⌀12mm carbon fiber tube that serves as the rear axle. It has the same mounting patter as the Main gear (6mm spacing between the 2 M3 screws). The Cup extends the 12mm cylinder outward to house the 12mm rod. It also has $2 \times$ ⌀2mm holes at each end to interface with the axle and transfer the load.

For printing, the cup is split into 2 parts axially (see Figure 40) and then printed flat to ensure the force moves along the layers and ensure ease of printability. It is printed at a layer height of 0.12mm, has 3 perimeters and a 25% infill. A brim is used to aid with bed adhesion.

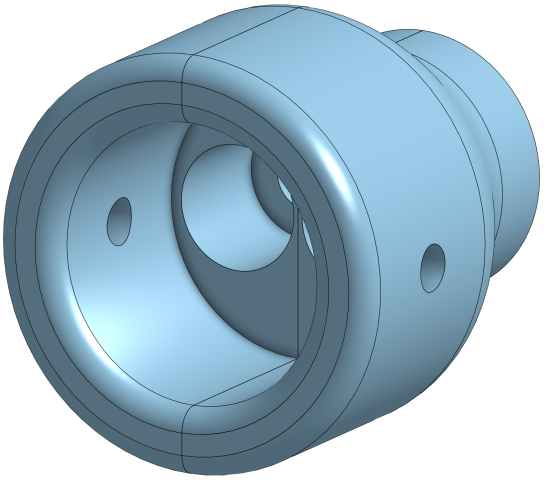


Figure 40: The Attaching Cup.

The Spacer

When looking at Figure 36, one can see that there are bearings on the outside as well, excluding the ones inside the housing. These are also 6901ZZ bearings and are responsible for the rotation of the housing itself. They are clamped from either end (discussed later). To ensure that there is →

1. No play in the mounting of the differential gearbox
2. Adequate separation between the bearing ends and the housing of the wall
3. Ease of motion

A spacer disk is mounted in between the two bearings. This disk also provides the pre-load needed in the gearbox stackup. It also ensures that the main gears in the differential mesh properly with the spider gears. Unlike the TPU spacer (Figure 35), this is made out of PLA.

It is printed at a layer height of 0.12mm, has 3 perimeters and a 25% infill. A brim is used to aid with bed adhesion. The height of the disk, nominally 2.00mm can be adjusted to adjust the pre-load. The grove of the roller bearing (Figure 10). The disk has an outer diameter of 16.40mm and an inner diameter of 12.4mm. You can see the orange disk in Figure 41 doing its job.

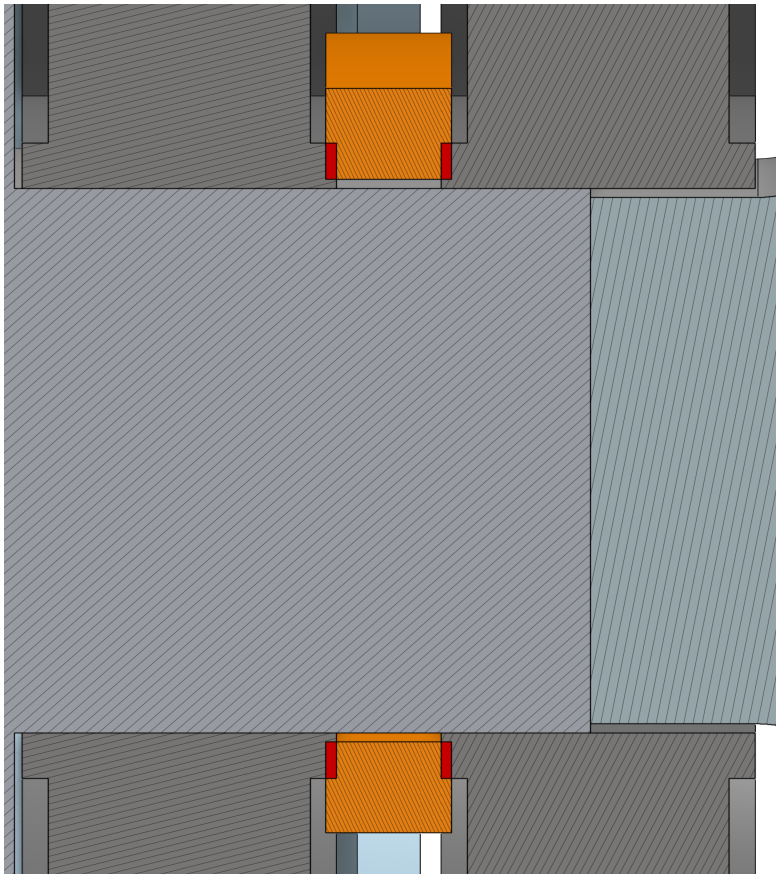


Figure 41: The Spacer Disk.

3.4 The Rear Axle

The rear axle is what really brings everything together. The mounts for the rear axle hold the motor, the batteries, the needle roller bearings and the differential. They are also the end point of the 4 carbon fiber rods used to strengthen the vehicle. The parts for the rear axle are also among the largest parts.

The rear axle houses the 2 BCWN-T-d12-IR bearings (Figure 11). We start our journey with the Rear Axle Mount. It encompasses the whole rear axle and only consists of 2 parts that work in tandem.

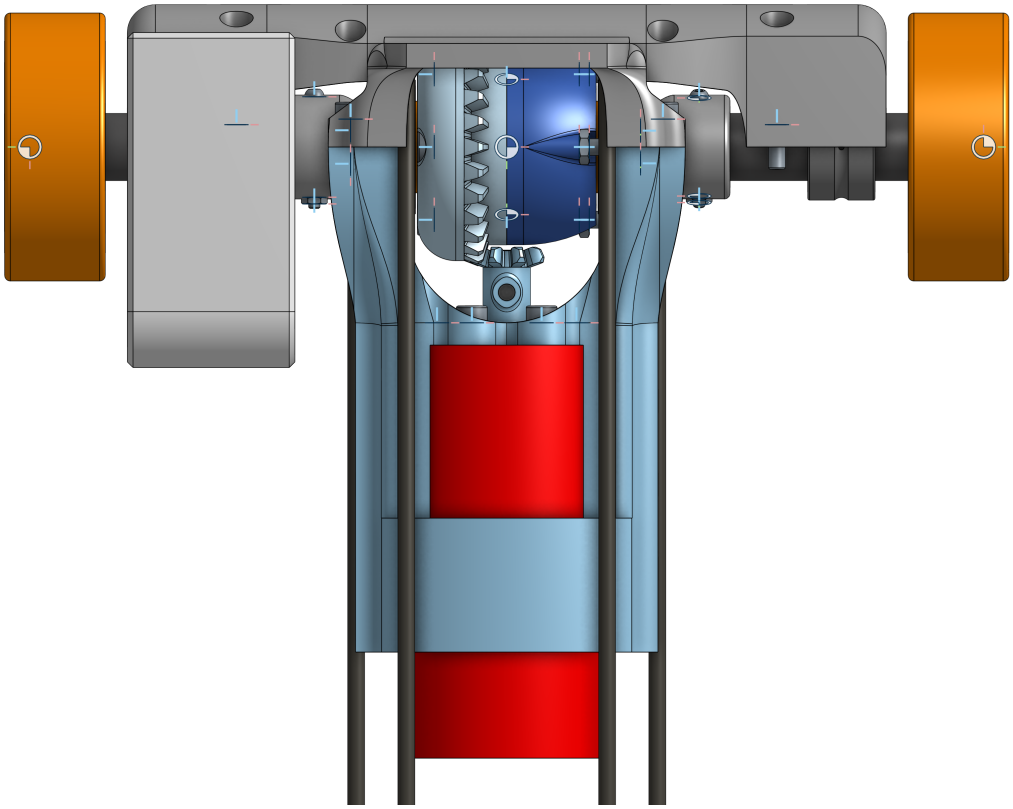


Figure 42: The Rear Axle Assembly.

Rear Axle Mount Front

The front mount is the light blue colored part in Figure 42. It houses the motor and serves as one half of the gearbox mount. It is one of the most complex parts.

It houses holes for 6 M3 × 8mm screws that connect the motor to the mount. It also has space for 4 M3 × 4mm heat set inserts at the very end to interface with the back mount and space for 6 M3 × 6mm heat set inserts, 3 on each side (see Figure 43) that allow it to interface with the parts that connect the front and the rear axle. The front mount also serves as the end point of the bottom set of ⌀3mm rods.

The part is printed with the forward facing face in Figure 42 pointing down, as shown in Figure 44. The part does need supports at the area where the motor mounts. It is printed at a layer height of 0.20mm, has 3 perimeters and a 25% infill. A brim is used to aid with bed adhesion.

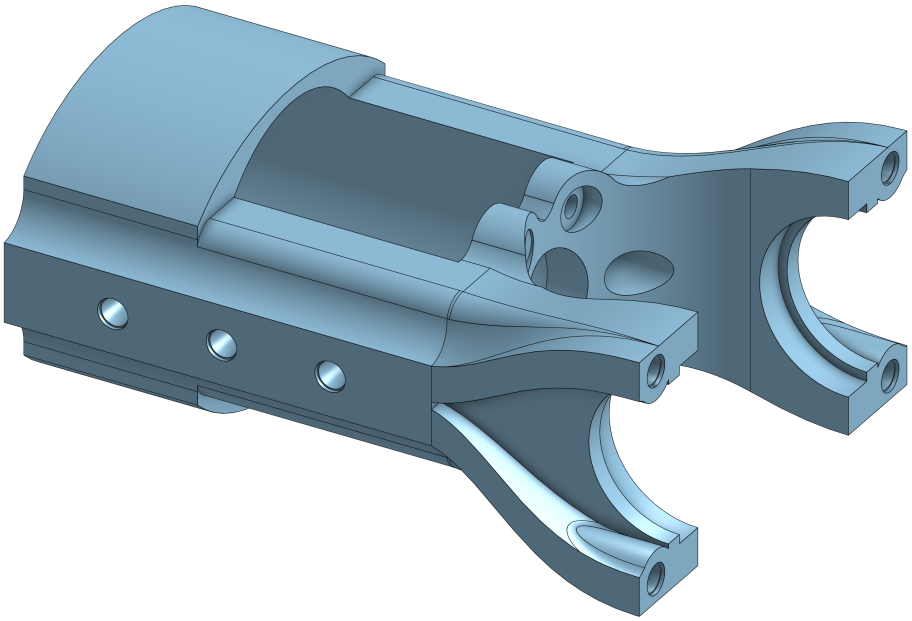


Figure 43: The Front Mount.



Figure 44: The Front Mount in the Slicer.

Rear Axle Mount Rear

The rear mount serves as a half to mount the BCWN-T-d12-IR as well as the 6901ZZ bearings. It's a fairly chunky piece. It uses 8 M3 × 8mm screws that interface with the rest of the vehicle. This part serves as the end mount of the top ⌀3mm carbon fiber rods.

The part prints with the front face facing downward. It is printed at a layer height of 0.20mm, has 3 perimeters and a 25% infill. A brim is used to aid with bed adhesion and prevent warping. Supports are not needed but appreciated.

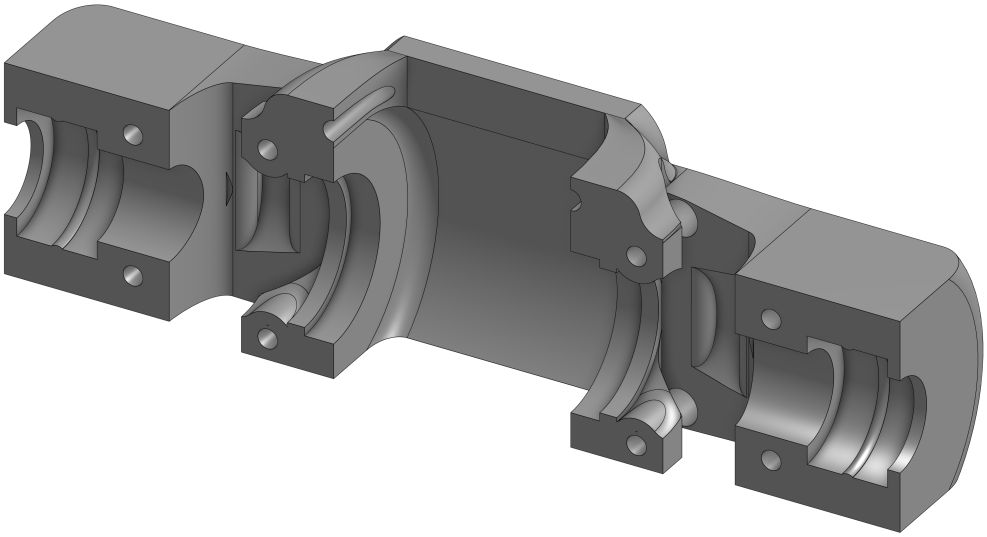


Figure 45: The Back Mount.

The batteries mount above the rear axle and are shown in the next section.

3.5 The Center

This part ties the the two axles together. It houses the LiDAR, the Raspberry Pi, the ESC and the batteries.

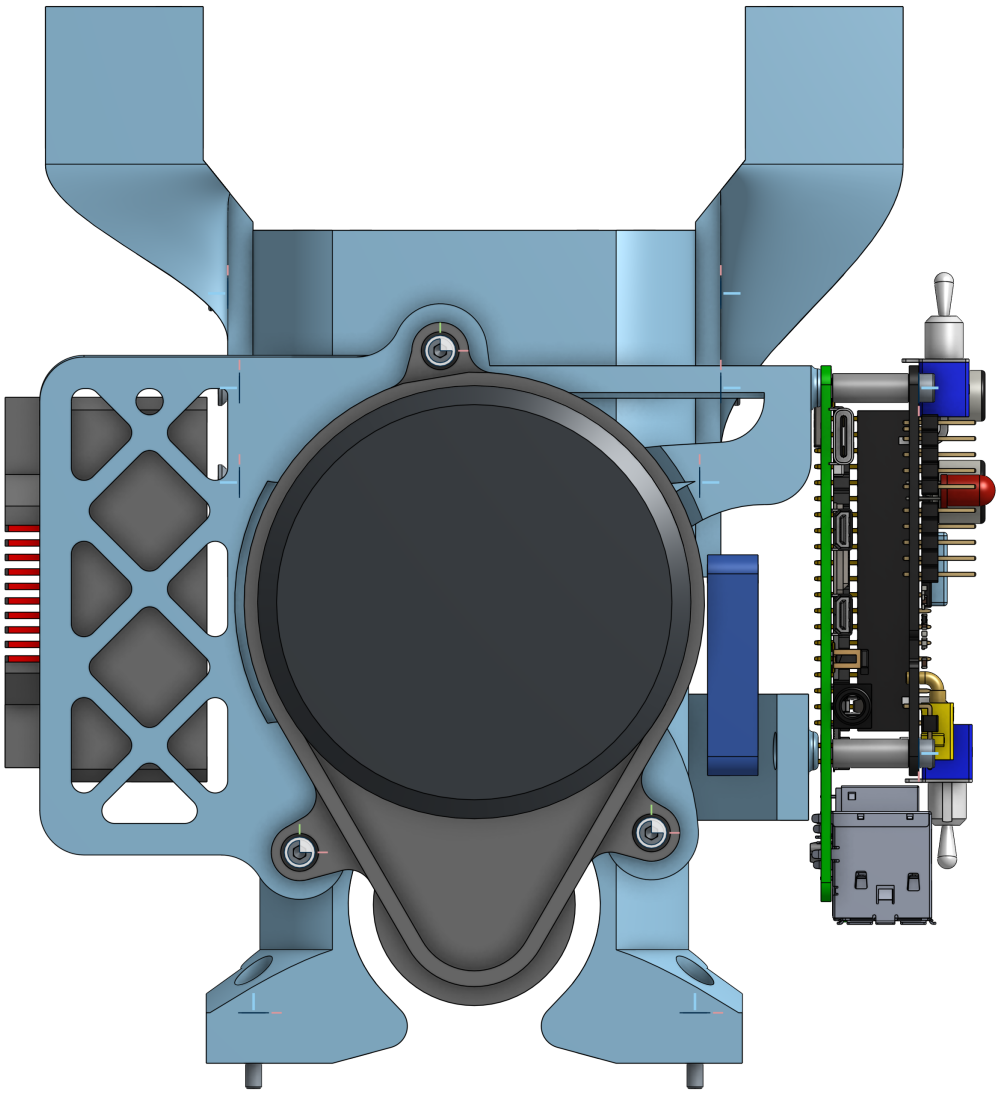


Figure 46: The Center Assembly.

Top Brace

The top brace is the main part that connects the motor mount to the steering column. The top supporting rods run directly, all the way through this. It is one of the two biggest parts of the entire vehicle.

It connects to the motor mount (Rear axle mount front) with 6 M3mm screws and connects with the steering mount with 2 M3 $\times$ 12mm screws. It takes 2 M3 $\times$ 6m heat set inserts on the top surface for the LiDAR & ESC mount.

The part is printed with the rear facing surface pointing downward (the part is upright) and does not require any supports. The holes at the front in Figure 47 are counter bore, but features are provided to help with printing. PreFlight can also be used to aid with counter bore bridging.

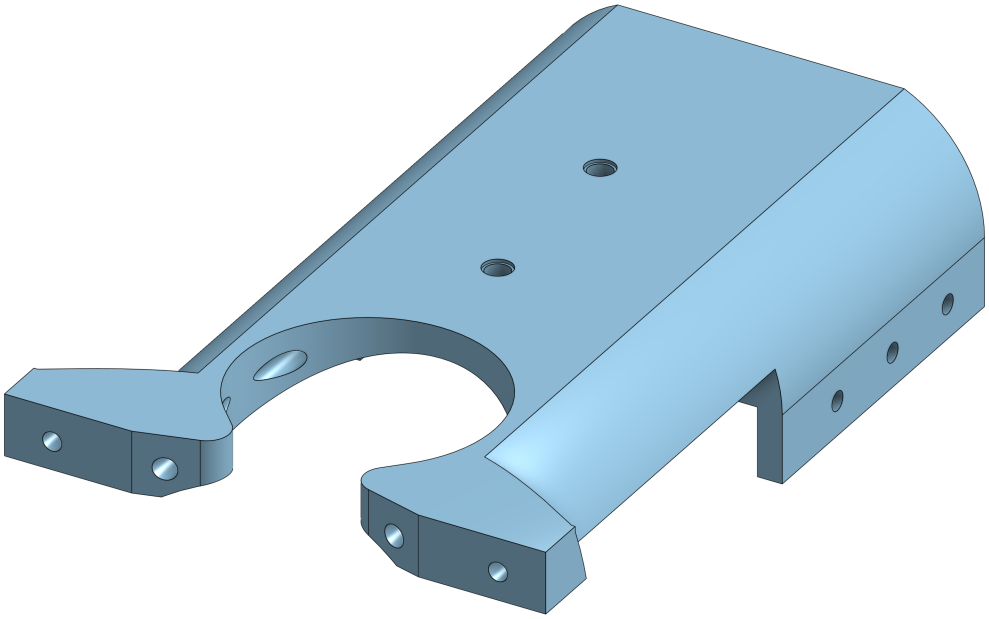


Figure 47: The Top Brace.

The part is printed at a layer height of 0.20mm, has 3 perimeters and a 25% infill. A brim is used to aid with bed adhesion and prevent the part from peeling off when printing.

Bottom Left & Bottom Right Brace

The bottom left and bottom right braces are just mirrored parts. They are tiny in comparison to the top brace. They help with connecting the top brace to the bottom pair of carbon fiber rods. They also interface with the motor mount (Rear axle mount front) on the same 6 points as the top brace. They are wedged between the motor mount and the top brace.

They are printed with either of the end, flat, sides pointing down. They do not need supports and are printed at a layer height of 0.20mm, has 3 perimeters and a 25% infill. A brim is used to aid with bed adhesion and prevent the part from peeling off when printing.

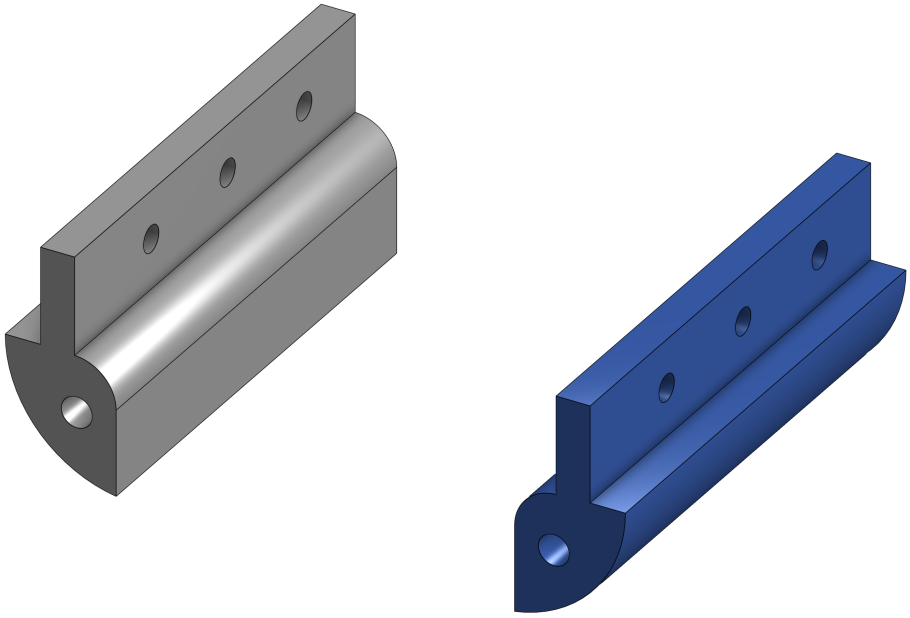


Figure 48: The Bottom Left and Right Brace.

Rear Bearing Mounts

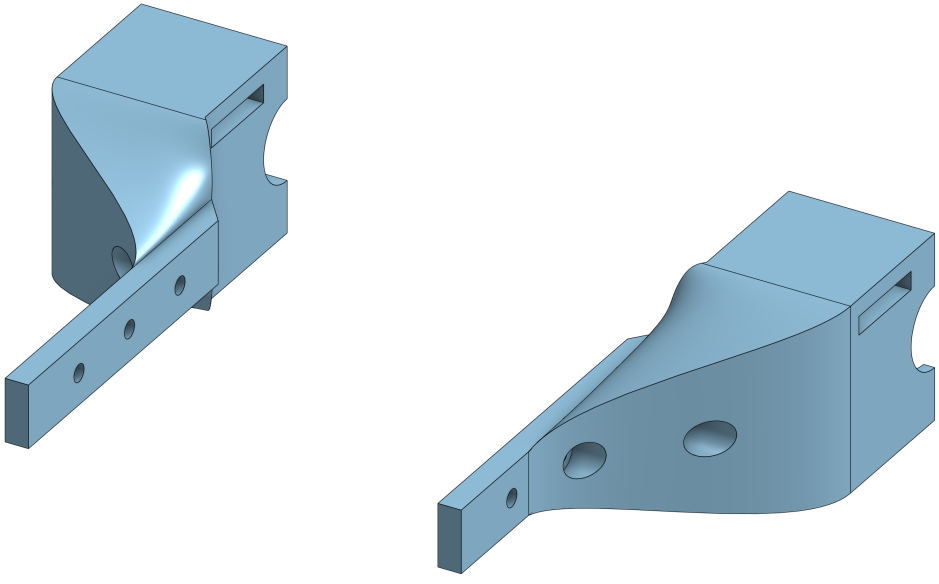


Figure 49: The Left and Right Bearing mounts.

These parts go on the outside of the motor mount and top brace sandwich. They serve 2 purposes, one of them is holding the BCWN-T-d12-IR bearing and the other is holding the batteries. The battery straps go though this part. Each one of them is more or less a mirror but the right one has a larger loft for no apparent reason.

They are printed with the bottom side facing down. They do not need supports and are printed at a layer height of 0.20mm, has 3 perimeters and a 25% infill. A brim is used to aid with bed adhesion and prevent the part from peeling off when printing.

LiDAR & ESC mount

The LiDAR and ESC mount is one of the two largest parts on the entire vehicle. They mount spans almost the entire breadth of the vehicle. It mouns the ESC on one side, screws into the top brace, mounts the LiDAR and connects to one of the bottom screw holes of the Raspberry Pi.

The part has holes to reduce its weight and a recess in on the top to accommodate for the LiDAR's electronics. The ESC has its fan removed and the screws from the shroud are used to mount. There are 3 M3 $\times$ 6mm heat set inserts to mount the LiDAR. The LiDAR uses 3 M3 $\times$ 10 countersunk screws. The mount prints on the face where the ESC mounts from. The part doesn't not need supports, but they are appreciated. It is printed at a layer height of 0.20mm, has 3 perimeters and a 25% infill. A brim is used to aid with bed adhesion and prevent the part from peeling off when printing.

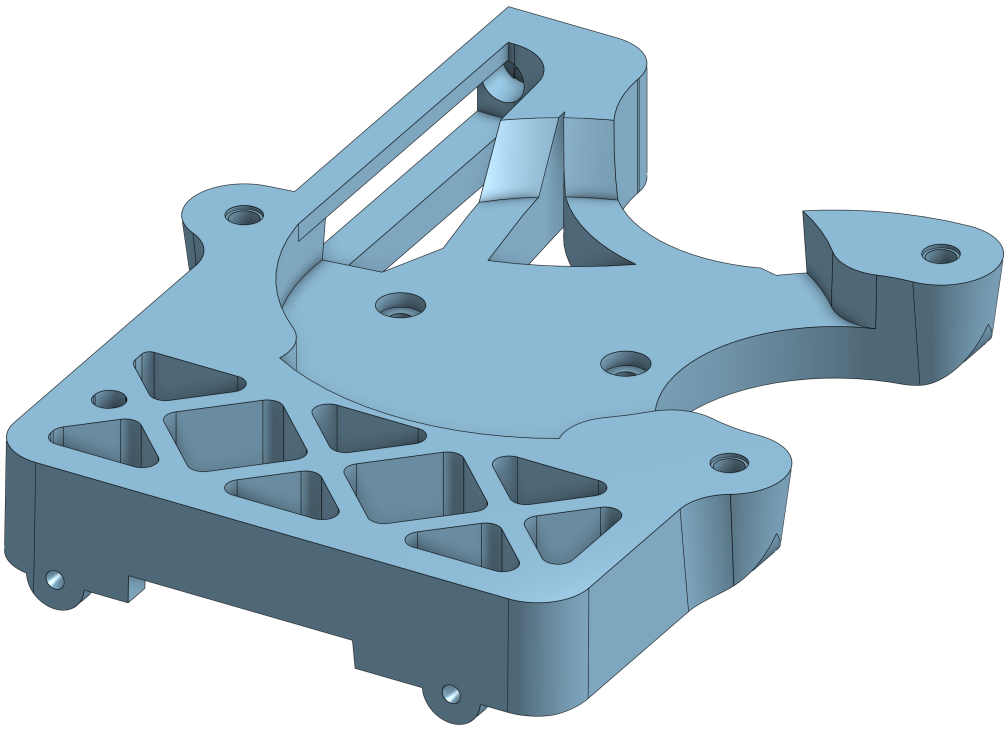


Figure 50: The Lidar & ESC mount.

3.6 Accessories

There are several parts that don't fall into a single category but are essential or quality of life improvements.

Wheel Hub

The wheel hub is the part that goes around the custom silicone casting as discussed earlier. It has to grip the cast tire and attach to the axles securely. To do this, it uses a single M2 $\times$ 12mm screw. The wheel hub is a tight interference fit with both the wheel and the axle. It is 18mm wide, same as the cast tire.

The wheel hub is printed at a layer height of 0.20mm has 3 perimeters and a 25% infill. A brim is used to aid with bed adhesion and prevent the part from peeling off when printing. The outer surface (where the wheel hub and tire touch), see Figure 51, has a fuzzy skin, with a thickness of 0.1mm and 0.2mm.

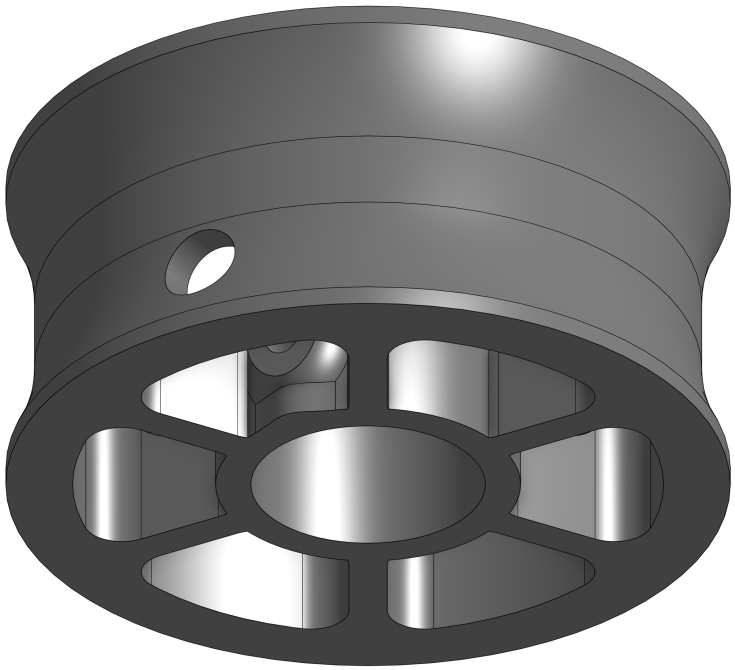


Figure 51: The Wheel Hub.

Raspberry Pi Mount

This part holds the raspberry pi by the forward 2 mounting holes and constrains it via attaching it to the bottom 2 carbon fiber brace rods. The part is fairly simple. It interfaces with the ⌀3mm rods via holes and interfaces with the pi with 2 M2.5 × 8mm screws.

The part is printed with any of the large, flat faces (see Figure 52) facing down and does not need supports. It is printed at a layer height of 0.20mm, has 3 perimeters and a 25% infill.

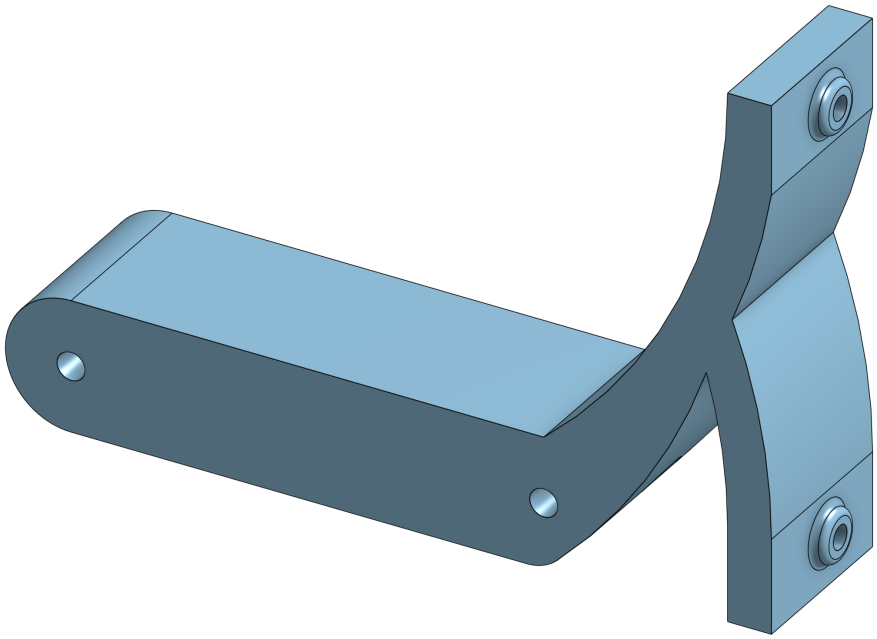


Figure 52: The Raspberry Pi Mount.

ESC Wire Mount

This is a very simple part that attaches with the bottom right brace rod and uses zip-ties to helpd the motor and ESC wires in a neat bundle. It's only 2mm think and serves the purpose. The part is printed with any of the large, flat faces (see Figure 53) facing down and does not need supports. It is printed at a layer height of 0.20mm, has 3 perimeters and a 25% infill.

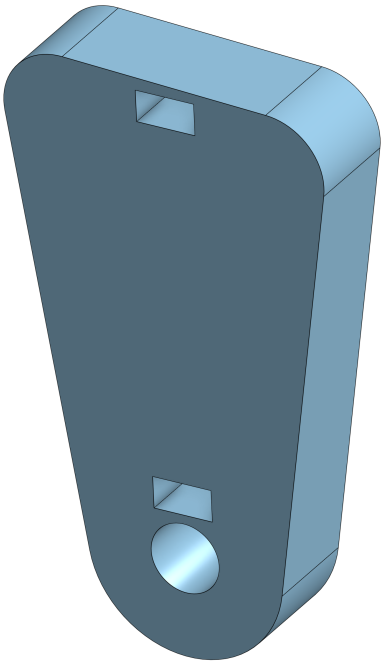


Figure 53: The ESC Wire Mount.

3.7 Tools

There were several tools used for cutting, drilling and casting. These tools don't make it to the final vehicle but are an integral part of the buld process.

Tire Molds

The mold for the tire, as mentioned earlier, is printed in Resin via the SLA manufacturing process. It is a two part mold that clamps via 4 M3 $\times$ 30mm screws and M3 nuts. It is advised to add washers to both sides. The mold does not have any draft angles as it was intended to be resin printed where the layer lines are microscopic and can be filled in by mold release. The mold can be FDM printed but would need draft angles.

It is a simple 2 par mold that has the negatives for the tire. The top and bottom are more or less mirrors of each other with the only difference being that the top has an entry and exit hole to help with the pouring process.

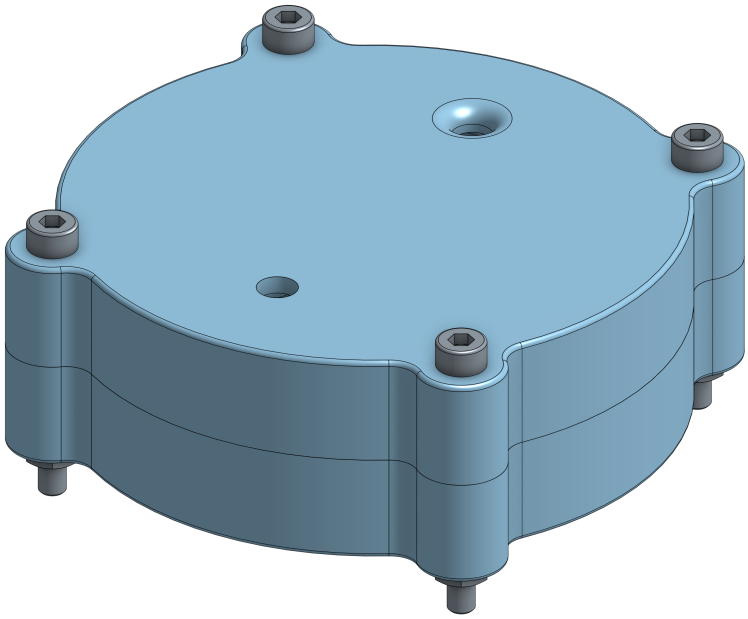


Figure 54: The Mold for the Polyurethane Tire.

Standoff Cutting

To attach the Raspberry Pi to the custom hat, 15mm nylon female to female standoffs are used. However, the actual spacing comes out to be 14mm. For a better fit, we decided to cut the standoffs to length, **however, this is completely optional**. To do this, a simple tool was made.

The part has the space for the standoff where it is pressed against a wall. It has a slit for a razor blade that allows it to slot in and cut straight and at length. The part is printed upright, as depicted in Figure 55. It is printed at a layer height of 0.20mm, has 3 perimeters and a 25% infill.

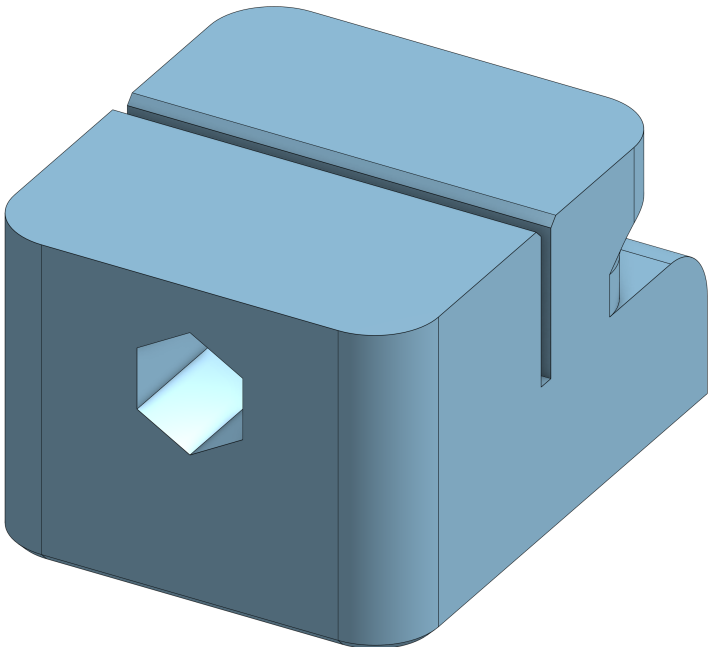


Figure 55: The Standoff Cutting Tool.

Axle Cutting

Caution: *Carbon Fiber dust is not very nice to inhale, all carbon fiber related tasks that involve manufacturing should be done with the part submerged in water to prevent the dust getting in the air and in a well ventilated area with a respirator and eye protection.*

The rear axle consists of 2 $\varnothing 12 \times 60$mm Carbon fiber tubes. The tube sourced was 500mm in length. To cut it down to length, a tool was needed.

The part takes in the carbon fiber tube until it is pressed against an inner wall. It has a slit for the saw blade. It is printed upright, as depicted in Figure 56. It is printed at a layer height of 0.20mm, has 3 perimeters and a 25% infill.

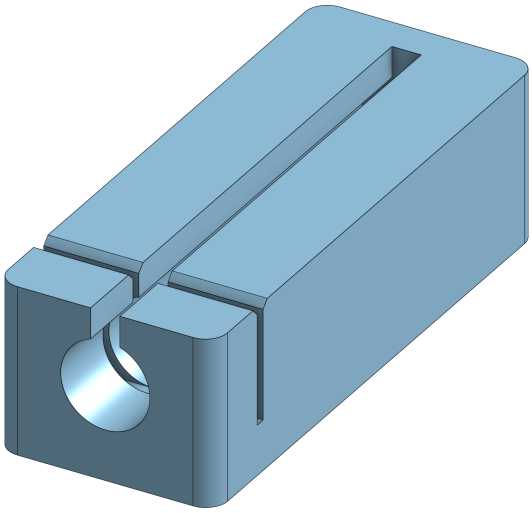


Figure 56: The Axle Cutting Tool.

Axle Drilling

Caution: *Carbon Fiber dust is not very nice to inhale, all carbon fiber related tasks that involve manufacturing should be done with the part submerged in water to prevent the dust getting in the air and in a well ventilated area with a respirator and eye protection.*

The rear axle interfaces with the other parts via M2 screws. This means that it needs a ⌀2mm hole at each end. This hole is spaced 4.5mm from each edge. To do this, a tool was needed.

The tool has a hole for a 2mm High Speed Steel or Carbide Drill Bit and a hole for the tube to slot in till it hits an internal wall. It is printed with the back face, the only face without any features or fillets, pointing down (see Figure 57). It is printed at a layer height of 0.20mm, has 3 perimeters and a 25% infill.

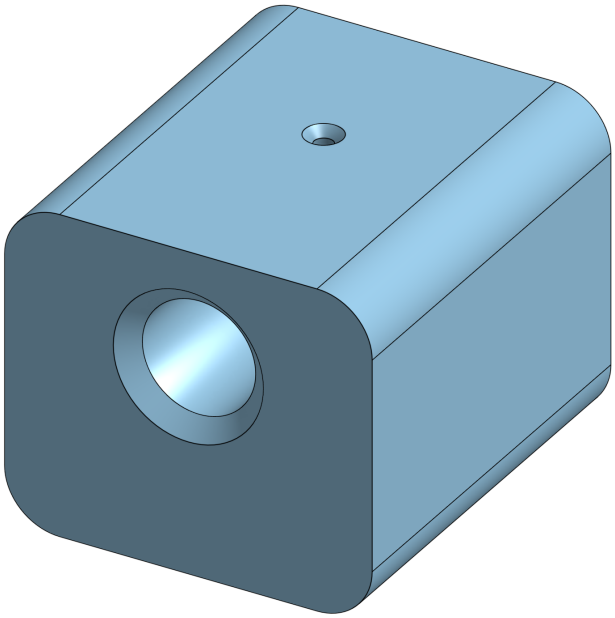


Figure 57: The Axle Drilling Tool.

3.8 Notes

All the parts are printed in PLA till mentioned otherwise.

The nominal settings are using the arachne wall engine with 3 bottom layers and 4 top layers with Gyroid infill. The nominal perimeter width is 0.4mm. One may need some post processing if not using a well calibrated printer. Some parts intentionally do not have clearances to interact better.

All the 3D files can be found in the "/models" directory, organized into sub sections. The files are in the ".STEP" format, allowing others to modify and change the files. The files are published under the GNU Affero General Public License (AGPL), meaning all changed files need to be published as well.

Thanks again to the Orca Slicer and the PreFlight Slicer teams for the amazing and open source software they provide.

The BOM for the screws is available in the "other" as well as "models" folder.

4 OS and Configurations

The Raspberry Pi uses a 64GB SD card. It is flashed with Ubuntu 22.04.5 Server. This is used with ROS 2 Humble Hawksbill. What is ROS and why use it?

ROS (Robot Operating System) is an open-source robotics middleware framework. Rather than a traditional operating system like Linux or Windows, ROS acts as a communication backbone. It connects the main compute board (e.g., Raspberry Pi) to your sensors (LiDAR, camera, IMU) and motor controllers, allowing different software modules to send data to each other seamlessly.

Reasons to use ROS

1. **Process Decoupling**: Runs perception, navigation, and motor control as separate, independent nodes. A crash or lag in image processing won't freeze the real-time steering/drive loop.
2. **Automated Frame Transformations (tf2)**: Computes precise spatial math and coordinate transforms between different physical points on the car (e.g., center of mass → LiDAR offset → camera offset).
3. **Sensor Synchronization**: Provides built-in tools (message filters) to align high-bandwidth feeds (camera frames and 2D LiDAR scans) with low-level IMU and encoder data.
4. **Data Logging & Offline Replay (rosbag)**: Records full track run data to replay offline, letting you refine algorithms and tune controllers without physical track time or draining LiPo batteries.
5. **Live Visualization (RViz)**: Displays real-time 2D LiDAR sweeps, camera overlays, and projected path trajectories live on screen during testing.
6. **WRO Rubric Alignment**: Fulfills competition evaluation criteria for software modularity, state-machine design, and code reproducibility required in the engineering journal and GitHub repository (refer to README.md).

Hence ROS was used. Why Humble Hawksbill? Because it was the latest stable release at the time of starting this.

4.1 Camera ROS 2 Integration

Ubuntu has a lot of driver issues, especially when running it on an SBC like a Raspberry Pi. They are annoying but navigable.

Driver Constraints & Hardware Sensor Binning

Configuring the Arducam IMX219 (175° FOV) on Ubuntu 22.04 LTS (64-bit) requires modern `libcamera` overlays rather than legacy MMAL drivers. To maintain the full field-of-view without cropping or high CPU overhead, the sensor's native **hardware binning (subsampling)** mode is utilized. This mode samples the entire 3280×2464 sensor plane but combines pixels on-chip down to 640×480, preserving the full 175° FOV while drastically reducing transport latency for real-time ROS 2 inference.

Configuration & Node Execution

Configure the sensor overlay in `/boot/firmware/config.txt` (`start_x=1`, `gpu_mem=128`, `dtoverlay=imx21` and reboot.

After installing `ros-humble-camera-ros`, launch the camera node in binned 640 × 480 `RGB888` format to publish to image/raw:

```
ros2 run camera_ros camera_node --ros-args \
  -p format:="RGB888" -p width:=640 -p height:=480
```

4.2 Hardware PWM Configuration & Tuning

Pinout & Driver Setup

Dual-channel hardware PWM on the Raspberry Pi 4B utilizes PWM0 (GPIO 12 / Pin 32) and PWM1 (GPIO 13 / Pin 33) operating at 50 Hz (20 ms period). Because the PWM hardware clock controller shares resources with onboard audio, audio must be disabled in firmware and blacklisted in the kernel:

1. **Firmware Overlay:** Add `dtoverlay=pwm-2chan,pin=12,func=4,pin2=13,func2=4` and `dtparam=audio=off` to `/boot/firmware/config.txt`.
2. **Kernel Blacklisting:** Blacklist the audio module by adding `blacklist snd_bcm2835` to `/etc/modprobe.d/raspi-blacklist.conf`.

Servo Pulse Limits & Power Stability

The S2500M steering servo is calibrated at 50 Hz with strict pulse boundaries:

- **Pulse Boundaries:** Neutral at 1500 μs, minimum safe pulse at 700 μs ($\approx 0°$), and maximum safe pulse at 2300 μs ($\approx 180°$).
- **Thermal/Current Throttling:** Exceeding 2300 μs forces the servo against mechanical hard stops, triggering internal thermal throttling with 5-second recovery delays.
- **Power Integrity:** Requires a common ground path across the Pi, HAT, and power rail, along with a 470 μF decoupling capacitor across 5V/GND to absorb inrush spikes.

Linux Sysfs Execution Sequence

When interfacing via `/sys/class/pwm/pwmchip0/`, write sysfs parameters in strict order:

1. Disable output (`enable = 0`).
2. Set period first (`period = 20000000`).
3. Set pulse width second (`duty_cycle = 1500000`).
4. Enable output (`enable = 1`).

5 Control Architecture & Software System

5.1 System Architecture & ROS 2 Pipeline

The software stack is structured as a modular ROS 2 architecture comprising three primary subsystems: Perception, Decision & Control, and Mobility/Actuation. Inter-node communication relies on standard ROS 2 topics to maintain real-time responsiveness.

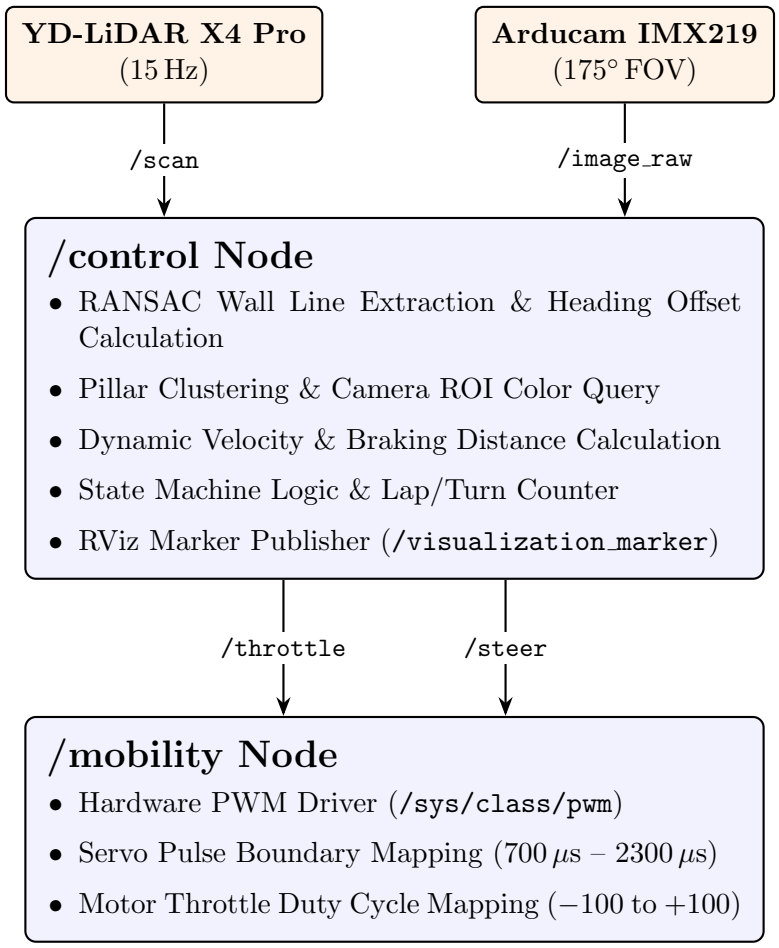


Figure 58: ROS 2 System Control Architecture and Inter-Node Dataflow Diagram.

5.2 ROS 2 Topics & Interfaces

- `/scan` **Topic (**`sensor_msgs/msg/LaserScan`**):** Published by the LiDAR node at a nominal frequency of 15 Hz, containing 2D range arrays over 360°.
- `/steer` **Topic (**`std_msgs/msg/Float32`**):** Published by the `/control` node to command target Ackermann steering angles. The `/mobility` node maps these commands to PWM duty cycles (700 μs to 2300 μs) on GPIO 12.
- `/throttle` **Topic (**`std_msgs/msg/Int8`**):** Published by the `/control` node to set motor velocity demands from −100 (full braking / reverse) to +100 (maximum forward acceleration), with 0 representing neutral.

- `/visualization_marker` **Topic (**`visualization_msgs/msg/Marker`**):** Streams 3D geometric markers to RViz to display vehicle position, extracted wall vectors, and mapped pillar locations.

5.3 Perception Algorithms & Path Extraction

5.3.1 RANSAC Wall Extraction & Heading Offset

2D range arrays from `/scan` within the perception radius are processed using a Random Sample Consensus (RANSAC) line-fitting algorithm to extract straight wall vectors. The controller calculates the lateral offset error e_d and heading angular error θ relative to the wall vector to derive the Ackermann steering command δ:

$$\delta = K_p \cdot \theta + K_d \cdot e_d \tag{1}$$

where K_p and K_d represent proportional and derivative control gains.

5.3.2 Dynamic Velocity-Dependent Braking

To prevent cornering overshoot, the braking initiation distance d_{brake} is calculated dynamically as a function of current vehicle velocity v:

$$d_{\text{brake}} = d_{\min} + k_v \cdot v^2 \tag{2}$$

where $d_{\min}$ is the baseline stopping distance and k_v is an empirical braking coefficient.

5.4 Challenge Execution Logic

5.4.1 Round One: Obstacle-Free Wall Following

Startup Sequence: Pressing the hardware start switch initiates a non-blocking startup delay. This allows the LiDAR motor to reach nominal operating speed (15 Hz) and ensures team members clear the operational area.

Wall-Following & Turn Management: The `/control` node evaluates wall vectors to execute directional turns, adjusts dynamic velocity/braking thresholds, and tracks completed turns and laps.

5.4.2 Round Two: Pillar Mapping & Parallel Parking

Exit Maneuver & Direction Initialization: Operation begins with an automated exit sequence from the parallel parking bay, after which the primary circuit direction is established.

Pillar Clustering & Camera Color Fusion: The `/control` node clusters contiguous points in `/scan` data to detect cylindrical pillars, computing their range and bearing angle ϕ. The node queries the wide-angle camera for average color values (RGB/HSV) within the Region of Interest (ROI) corresponding to bearing ϕ. The identified color (Red/Green) is logged on the occupancy map to dictate navigation paths (e.g., passing Red pillars on the right, Green on the left).

Automated Parallel Parking Finalization: Upon completing the pillar circuit, the vehicle transitions into parallel parking mode, utilizing lateral wall range measurements to execute a multi-point alignment into the parking bay.

6 Code Architecture

The autonomous software stack is engineered as a distributed, modular ROS 2 (Humble Hawksbill) architecture executed natively on Ubuntu 22.04 LTS on the Raspberry Pi 4B. By decoupling high-rate hardware actuation from multi-sensor perception and asynchronous state machine logic, the vehicle achieves deterministic 30 Hz control cycles with sub-millisecond inter-process transport latencies.

6.1 Round 1: Obstacle-Free Track Traversal

The primary objective of Round 1 is to complete three full laps (12 consecutive 90° turns) in either Clockwise (CW) or Counter-Clockwise (CCW) direction around the closed 1000 mm wide hallway, maintaining high cruise velocities and terminating smoothly within the start/finish zone.

Node Topology & Dataflow Pipeline

The Round 1 software stack consists of four core microservices communicating over ROS 2 topics using Best-Effort QoS profiles:

- `ydlidar_ros2_driver`: Acquires 2D range arrays over 360° at 15 Hz, publishing raw distance arrays to `/scan`.
- `wro_control_node`: The central intelligence node executing LiDAR sector decomposition, outer wall heading estimation, Hallway-Aligned coordinate projection, and the 5-State Finite State Machine (FSM) at 30 Hz.
- `mobility_node`: Hardware abstraction node interfacing directly with Linux sysfs PWM drivers (`/sys/class/pwm/`), mapping normalized steering demands ($\delta \in [-0.75, +0.75]$) and motor throttle commands ($P \in [-100, +100]$) to microsecond hardware pulses on GPIO 12 and GPIO 13.
- `start_switch_node`: Hardware interrupt monitor on GPIO 17 with debounced push-button edge detection, transmitting high-priority start triggers over `/start_trigger`.

Perception Geometry & Coordinate Transformations

Raw 360° point clouds from `/scan` are partitioned into spatial angular sectors:

- **Front Sector ($\theta \in [-25°, +25°]$):** Computes the 15th percentile distance d_{front} to detect upcoming 90° arena walls while filtering transient laser noise.
- **Left Sector ($\theta \in [+65°, +115°]$):** Computes the median orthogonal distance d_{left} to the left perimeter wall.
- **Right Sector ($\theta \in [-115°, -65°]$):** Computes the median orthogonal distance d_{right} to the right perimeter wall.

When navigating sharp turns or non-zero vehicle yaw angles, raw forward range measurements suffer from perspective foreshortening against the inner corner wall. To prevent premature or false turn triggers, points are transformed into the **Hallway-Aligned Coordinate Frame** (x', y') using the instantaneous outer-wall heading angle ψ:

$$\begin{bmatrix} x' \\ y' \end{bmatrix} = \begin{bmatrix} \cos\psi & -\sin\psi \\ \sin\psi & \cos\psi \end{bmatrix} \begin{bmatrix} x \\ y \end{bmatrix} \tag{3}$$

This projection rotates the LiDAR point cloud along the true longitudinal hallway vector, isolating the front wall at distance x' and excluding inner wall returns from the longitudinal distance metric.

Outer Wall Heading Estimation

The vehicle heading angle relative to the outer boundary wall is estimated in real time using fast analytical least-squares linear regression over the outer wall sector:

$$\psi = -\arctan\left(\frac{dy}{dx}\right) = -\arctan\left(\frac{S_{xy}}{S_{xx}}\right) \tag{4}$$

where $S_{xx} = \sum(x_i - \bar{x})^2$ and $S_{xy} = \sum(x_i - \bar{x})(y_i - \bar{y})$. A positive heading angle $\psi > 0$ indicates that the vehicle's nose is angled inward toward the center track wall.

Dual-Mode Lateral Centering PID with Heading Damping

The steering command $\delta(t)$ is generated via a composite PD controller coupled with an active heading damping term:

$$\delta(t) = K_{\text{lat}} \cdot e_{\text{lat}}(t) + K_{d,\text{lat}} \cdot \frac{de_{\text{lat}}(t)}{dt} - K_{\text{head}} \cdot \psi(t) \tag{5}$$

where the lateral error $e_{\text{lat}}(t)$ is evaluated based on wall visibility modes:

- **Symmetrical Centering Mode (Both Walls Visible):**

$$e_{\text{lat}} = \frac{d_{\text{left}} - d_{\text{right}}}{2} \tag{6}$$

- **Single Outer-Wall Tracking Mode (Corner Apex / Outer Wall Following):**

$$e_{\text{lat}} = \begin{cases} d_{\text{target}} - d_{\text{right}}, & \text{CCW (Outer Wall on Right)} \\ d_{\text{left}} - d_{\text{target}}, & \text{CW (Outer Wall on Left)} \end{cases} \tag{7}$$

The derivative term is filtered through a low-pass first-order exponential smoothing filter ($\alpha = 0.40$) to attenuate high-frequency differentiation noise:

$$\dot{e}_{\text{filt}}(k) = \alpha \cdot \frac{e_{\text{lat}}(k) - e_{\text{lat}}(k-1)}{\Delta t} + (1 - \alpha) \cdot \dot{e}_{\text{filt}}(k-1) \tag{8}$$

Finite State Machine (FSM) Execution Flow

Mission progression is governed by a deterministic 5-State Finite State Machine (Figure 59):

1. `WAIT_FOR_START`: Zero throttle, neutral steering. Awaits push-button trigger or ROS 2 `/start_trigger` event.
2. `STARTUP`: Executes a straight neutral launch ($\delta = 0.0$, throttle $= 25$) for 0.5 s. This bypasses the centering PID on the start line to eliminate servo hunting before clearing the start box.
3. `CRUISING`: High-speed hallway tracking at target cruise throttle ($P_{\text{cruise}} = 45$). The controller monitors d_{front} and transitions to `TURNING` when $d_{\text{front}} < d_{\text{turn\_thresh}}$ (0.78 m) and inner wall distance exceeds the collision lockout safety boundary ($d_{\text{inner}} > 0.35$ m).

4. `TURNING`: Asserts maximum steering lock ($\delta = -0.75$ for CCW, $+0.75$ for CW) at reduced cornering throttle ($P_{\text{turn}} = 30$). The vehicle exits turn mode when the forward corridor clears ($d_{\text{front}} \geq 1.25\,\text{m}$) or upon reaching the safety timeout cap ($t_{\text{turn}} \geq 0.85\,\text{s}$). The turn counter increments ($N_{\text{turns}} \leftarrow N_{\text{turns}} + 1$).

5. `FINISH_ROLLOUT` $\rightarrow$ `FINISH`: Upon completing Turn 12, the state machine transitions to `FINISH_ROLLOUT`, cruising down the home straight for 1.2 s into the designated finish box before commanding throttle 0 with an active 0.15 s reverse braking pulse in `FINISH`.

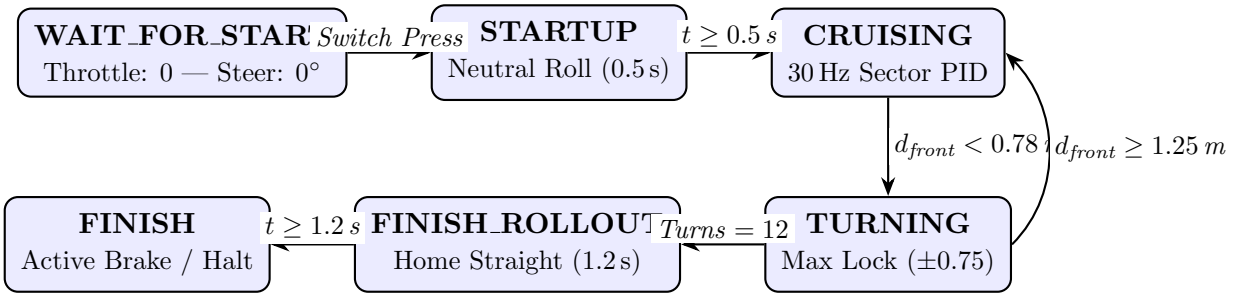


Figure 59: Round 1 Finite State Machine (FSM) Execution Flow.

Table 2: Round 1 Controller Configuration and Tuning Parameters

Parameter	Value	Functional Description
K_{lat} **(Lateral Proportional)**	0.95	Proportional gain for lateral corridor error correction.
$K_{d,\text{lat}}$ **(Lateral Derivative)**	0.15	Derivative gain for rate-of-error damping.
K_{head} **(Heading Gain)**	0.015	Angular dampening coefficient to prevent hallway oscillation.
$d_{\text{turn\_thresh}}$ **(Turn Trigger)**	0.78 m	Forward distance threshold for initiating 90° turn sequence.
$d_{\text{exit\_thresh}}$ **(Exit Clearance)**	1.25 m	Cleared hallway distance required to exit turn mode.
$t_{\text{turn\_timeout}}$ **(Turn Safety Cap)**	0.85 s	Maximum time permitted in turn state to prevent overturning.
d_{lockout} **(Inner Lockout)**	0.35 m	Minimum distance to inner wall before permitting turn entry.
t_{rollout} **(Rollout Duration)**	1.20 s	Straight cruise duration to position vehicle inside finish box.

6.2 Round 2: Obstacle Avoidance & Vision-LiDAR Fusion

Round 2 introduces dynamic 50×50 mm colored obstacle pillars randomly distributed across the track corridors. Vehicles must adhere to competition traffic rules—passing **Red pillars on the right** and **Green pillars on the left**—while negotiating 12 corner turns and executing an autonomous parallel parking maneuver upon race completion.

Multi-Node System Architecture

To achieve high-speed obstacle evasion without compute-induced latency, Round 2 divides perceptual tasks between spatial geometric processing and lightweight chrominance inference:

- `camera_node`**:** Captures 640×480 binned video streams at 30 Hz via the Arducam IMX219 ultra-wide 175° lens.
- `spike_detector_node` (`mvp_camera`)**:** An ultra-lightweight, zero-OpenCV color classification engine operating directly on vectorized NumPy memory buffers, publishing `"RED"`, `"GREEN"`, or `"CLEAR"` to `/camera_spike` at 3 Hz (reducing CPU load by $> 85\%$).
- `fast_wall_tracker_node`**:** Executes Hollow-Core spatial masking, horizontal RANSAC linear regression, and 85th-percentile longitudinal distance filtering on LiDAR scans, publishing `/wall_tracking_extended`.
- `reactive_drive_node` (`mvp_controller`)**:** Central decision and steering engine executing threat horizon gating, dynamic target offset generation, high-gain reactive dodging ($K_{\text{lat,dodge}} = 2.80$), and mission progression.

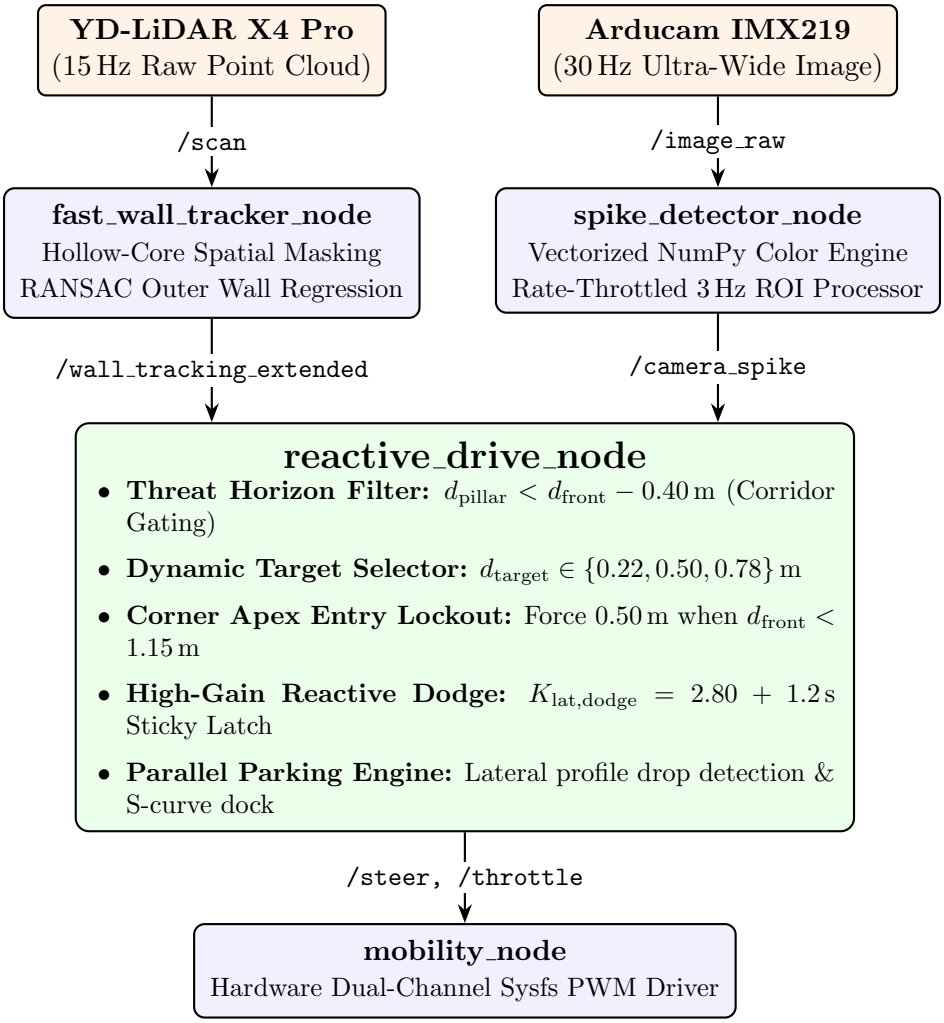


Figure 60: Round 2 Perception, Sensor Fusion, and Reactive Control Dataflow Pipeline.

Hollow-Core Spatial Masking & RANSAC Wall Fitting

In Round 2, physical 50 mm pillars located in the center corridor create false cluster spikes that corrupt naive wall estimation. To prevent this, `fast_wall_tracker_node` applies **Hollow-Core Spatial Masking**, filtering out all points in the lateral corridor center ($y \in [-0.35, +0.35]$ m relative to track midline) before performing wall regression.

The remaining boundary points are processed using a specialized horizontal RANSAC algorithm. The model hypothesizes line parameters $y = mx + c$ by enforcing a strict horizontal prior ($|dx| \geq 1.5|dy|$) and a point-to-line distance threshold of 45 mm:

$$d_i = \frac{|ax_i + by_i + c|}{\sqrt{a^2 + b^2}} \leq 0.045\,\text{m} \tag{9}$$

Inlier sets are refined through analytical least-squares regression, yielding true outer wall distance d_{outer} and heading ψ_{outer} completely immune to obstacle interference. The front wall distance d_{front} is extracted from the 85th percentile of forward sector ranges, discarding front-facing pillar points.

Zero-OpenCV Vectorized NumPy Spike Detector

Rather than executing resource-intensive object detection networks (e.g., YOLO), color identification is achieved via a vectorized NumPy chrominance evaluator. The algorithm crops the bottom 2/3 Region of Interest (ROI) and eliminates 15% margins on lateral edges to reject outer wall reflections.

The raw RGB byte buffer is subsampled by factor $S = 2$ and converted to normalized color selectivity ratios:

$$R_{\text{score}} = \frac{R}{G + B + 1}, \quad G_{\text{score}} = \frac{G}{R + B + 1} \tag{10}$$

Pixels satisfying $R_{\text{score}} > 1.45$ (with $R > 80$) are classified as Red, while pixels satisfying $G_{\text{score}} > 1.25$ (with $G > 70$) are classified as Green. If the active pixel count exceeds the noise threshold ($N_{\text{thresh}} = 80$ px), the corresponding classification is published to `/camera_spike`.

Threat Horizon Gating & Dynamic Offset Injection

A critical challenge in obstacle avoidance is preventing false dodges caused by background pillars located around the upcoming corner. The controller evaluates a strict **Threat Horizon Gate**:

$$\text{IsThreat}(d_{\text{pillar}}, d_{\text{front}}) = \begin{cases} \text{True}, & \text{if } 0.15\,\text{m} \leq d_{\text{pillar}} \leq 1.25\,\text{m} \quad \text{and} \quad d_{\text{pillar}} < (d_{\text{front}} - 0.40\,\text{m}) \\ \text{False}, & \text{otherwise} \end{cases} \tag{11}$$

When a valid threat is confirmed, the controller injects a dynamic target offset d_{target} relative to the outer wall based on drive direction (Table 3).

High-Gain Reactive Dodge & Sticky Latch Controller

Once an evasive offset is determined, the steering controller executes high-gain lateral correction:

$$\delta(t) = K_{\text{lat,dodge}} \cdot (d_{\text{target}} - d_{\text{outer}}) - K_{\text{head}} \cdot \psi_{\text{outer}} \tag{12}$$

Table 3: Dynamic Target Outer Wall Offsets for Obstacle Evasion

Drive Direction	Detected Spike	Target Outer Offset (d_{target})	Evasive Trajectory & Action
CCW (Outer on Right)	RED	0.22 m (Dodge Right)	Hug right outer wall to pass pillar on its right side.
CCW (Outer on Right)	GREEN	0.78 m (Dodge Left)	Hug left inner wall to pass pillar on its left side.
CW (Outer on Left)	RED	0.78 m (Dodge Right)	Shift toward right inner wall to pass pillar on right.
CW (Outer on Left)	GREEN	0.22 m (Dodge Left)	Shift toward left outer wall to pass pillar on left.
Any Direction	CLEAR	0.50 m (Corridor Center)	Maintain nominal track center line.
Approaching Corner	Any Spike ($d_{\text{front}} < 1.15$ m)	0.50 m (Forced Center)	Center vehicle for clean 90° corner apex entry.

with $K_{\text{lat,dodge}} = 2.80$. To ensure the vehicle fully traverses past the obstacle before returning to the centerline, the target offset is locked by a 1.20 s **Sticky Latch Timer**. A soft wall cushion boundary ($d_{\min} = 0.16$ m) guarantees that evasive steering never results in physical wall contact.

Automated Parallel Parking Protocol

Upon completing Turn 12, the vehicle transitions from circuit racing to parallel parking mode:

1. **Parking Bay Identification:** Scans the lateral LiDAR profile at 15 Hz to detect the rectangular opening (350 × 150 mm) indicated by a sharp step increase in lateral range ($d_{\text{lat}} > 0.85$ m).
2. **Positioning & Reversal Docking:** Advances 0.25 m past the opening, engages reverse throttle ($P = -25$), and executes a coordinated two-phase Ackermann S-curve trajectory ($\delta = +0.75 \rightarrow -0.75$) into the bay.
3. **Final Alignment & Lock:** Aligns flush with the back wall using longitudinal range checks and commands full brake ($P = 0$, $\delta = 0.0$), securing maximum score points.

7 Conclusion & Acknowledgments

Building our autonomous vehicle for the WRO Future Engineers challenge has been an invaluable journey of iterative design, engineering trade-offs, and practical problem-solving. From casting custom 40A polyurethane tires and designing a 4-layer PCB hat to configuring ROS 2 nodes and tuning hardware PWM timings, every sub-assembly represents hands-on learning and technical growth.

7.1 Document Production

This manual and technical documentation suite were authored using LaTeX and compiled using **TeXmaker**. In the spirit of open collaboration, all CAD STEP models, schematics, and hardware BOMs are published under the GNU Affero General Public License (AGPLv3) for the robotics community.

7.2 Open-Source Acknowledgments

This project relies heavily on the open-source ecosystem and accessible software platforms. We express our gratitude to the developers, maintainers, and communities behind the following tools:

- **TeXmaker & LaTeX:** For providing a reliable typesetting environment for technical manual generation.
- **KiCAD:** For enabling 4-layer PCB schematic capture, routing, and 3D visualization.
- **Orca Slicer & PreFlight Slicer:** For advanced toolpath generation, Arachne wall engines, and precise 3D printing execution.
- **ROS 2 (Humble Hawksbill):** For supplying a robust middleware framework for robotics perception and node communication.
- **Onshape:** For flexible 3D CAD modeling and parametric assembly management.
- **Interactive HTML BOM:** For generating interactive assembly guides for surface-mount soldering.
- **Linux & Raspberry Pi Community:** For maintaining the embedded drivers, overlays, and hardware documentation.

Thank you to the WRO organizers, judges, fellow competitors, and the broader open-source community for making this journey possible.