TEAM DARVIN

Design Report

SOCRATES

Faculty Mentor - Prof. Kashyap Joshi
(Electronics and Telecommunication Dept.)
kashyap.joshi@nmims.edu

Team Captain - Sarthak Mishra
sarthak.mishra25@nmims.edu.in

Date Submitted: 10th May 2022

Team Members
Vaibhav Raheja - Vice Captain - vaibhav.raheja34@nmims.edu.in
Apurv Gupta - Core Member - apurv.gupta09@nmims.edu.in
Vatsal Chanchpara - Core Member - vatsal.chanchpara04@nmims.edu.in
Nidhi Koppikar- Core Member - nidhi.koppikar35@nmims.edu.in

SVKM'S NMIMS Deemed to be UNIVERSITY | MUKESH PATEL SCHOOL OF TECHNOLOGY MANAGEMENT & ENGINEERING ™ TEAM DARVIN

Statement of Integrity

I, Professor Kashyap Joshi, hereby declare that the work done by Team D.A.R.V.I.N under my guidance for the IGVC competition 2022 has been significant and equivalent to what might be awarded credit in a senior design course.

[illegible]

Prof. Kashyap Joshi
EXTC Department
NMIMS MPSTME
(kashyap.joshi@nmims.edu)

Table of Contents

1 Team Organization

The members of Team Darvin have been working throughout the COVID Pandemic to master the fundamentals of engineering to perfect the robot for IGVC 2022. This year, we decided to work as a small team with members who had experience in all the core engineering fields. These members would contribute to complete the designing, manufacturing, and testing of the robotic vehicle.
The work for this project was divided into mainly 4 sections: Mechanical, Electronics, Software, Administrative.

Name	Position	Major work
Sarthak Mishra	Team Captain	Software, Electronics, Administrative
Vaibhav Raheja	Vice-Captain	Software, Mechanical
Apurv Gupta	Core Member	Software, Mechanical
Vatsal Chanchpara	Core Member	Electronics, Software
Nidhi Koppikar	Core Member	Electronics, Software

2 Design Process

The first step we took in designing Socrates is to study and separate the problem statement into blocks. Each team member was then assigned to a task that most suited their interest and major. We started by choosing the components required for a basic vehicle in order to test our software first. We would then upgrade these to improve our vehicle in order to qualify for IGVC. Brainstorming sessions were held at regular intervals in order to keep track of progress as well as to take into consideration new ideas and build upon.

3 Innovations

3.1 Mechanical

Our frame was designed after studying the chassis of various teams from previous years. Many of the designs were 'top heavy' due to their camera stands resulting in instability while driving. Our camera stand is securely mounted at a height which does not raise the center of gravity by a lot. It is kept simple in structure to speed up assembly time and is very modular due to the aluminum extrusions

3.2 Electrical

We have designed a custom PCB for bridging our computer system with the high-power motor drivers. This PCB was designed with an ATMEGA2560 and is based on the Arduino mega. A similar second PCB was designed for handling tasks such as the wireless kill functionality and the battery temperature management and diagnostics.

3.3 Software

For lane detection an image processing pipeline was created and altered as per our requirement. Object detection and avoidance was achieved with the help of Velodyne Lidar. Our vehicle is driven based on a target heading and speed. This is controlled by a PID loop and the feedback from the IMU. The GPS navigation uses the Euclidean distance math and finds the shortest path between 2 points. We also designed our vehicle to adjust its speed based on the steering angle so as to accurately navigate the course.

4 Mechanical Design

4.1 Chassis

Our chassis has been designed completely with 3030 aluminum extrusion. This provides modularity to our vehicle and is lightweight at the same time. Our chassis design is kept simple to avoid instability while driving. It is designed to enclose the payload and all other items within itself with the help of acrylic sheets. We have kept the camera stand 5ft above ground and it is modular for adjustable positions of the ZED camera.

4.2 Wheel Assembly Design

We have chosen to place the DC motors at the front for better control of the bot with a caster wheel at the back for turning. We are using 2 100W RCMS-2088 DC planetary geared motors running at 24v, each with a torque of 80 kgcm based on our load calculations which proved to be plenty during our testing. It can achieve a theoretical max loaded speed of 3.5mph. These motors are resting on a custom designed stainless steel motor mount, since our wheels did not have a shaft, a custom wheel hub is designed for connecting the motor shaft and wheel with a mild steel coupling.

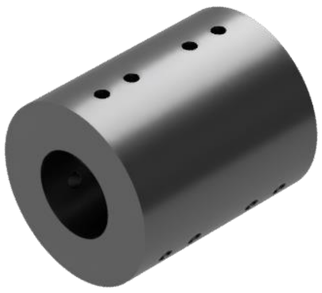
Fig 1: Coupling

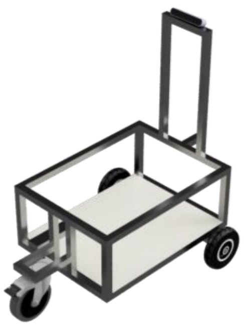
Fig 2: Bot with Wheel placement

4.3 Lidar (VLP-16) and Camera(ZED) Weather Protection

We are using a Velodyne VLP-16 as our lidar for object detection and a ZED Sterolabs camera for lane detection. For weather protection we have designed, and 3D printed a custom PETG cover for both these devices.

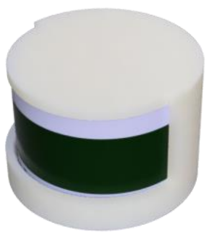
Fig 3: Velodyne Cover

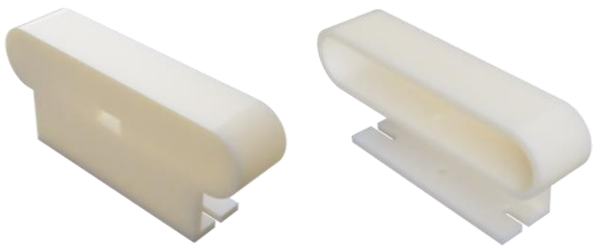
Fig 4 and Fig 5: Zed Camera Protection Cover

4.4 Encoder mount

We are using an external encoder to control and drive our motors accurately. We are using a 360 ppr and 1440 counts encoder. We have carefully designed and attached gears, to minimize backlash, between the encoder and motor output shaft to get a 1:1 gear ratio.

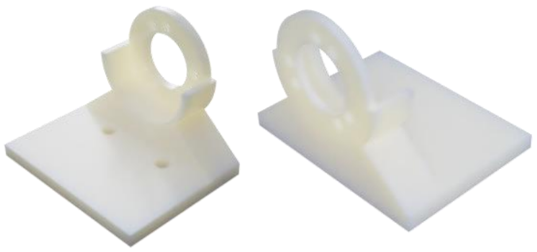

Fig 6 and 7: Encoder Mount

4.5 Battery Case

An enclosure for the battery was designed and manufactured by 3d printing in PETG. This enclosure contains the 2 batteries required to power our robot. It was also designed to provide hotswappability of the batteries. The front plate of the battery enclosure contains 2 XT60 connectors that can handle the high amperage flowing from the battery. We decided to put the female versions of the XT60 connectors on the battery as these have the housing plastic in between the 2 contacts. Hence the female versions prevent short circuits and easy connection and disconnection. Similarly, a fixed piece was 3d printed and contains the male versions of the XT60 connectors. This enclosure also contains a slot for a fan for managing the temperature of the batteries.

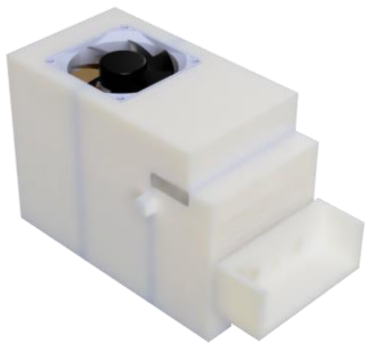

Fig 8: Battery Case

4.6 Wireless E-Kill Switch

A custom enclosure for housing the e kill, PCB and battery is designed and 3d printed. This casing is made waterproof and houses the entire circuit needed for the e kill.

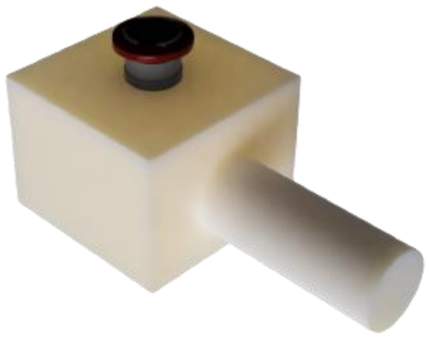

Fig 9: Wireless E-Kill Switch Housing

5 Electronic Design

5.1 Power delivery and Kill Switch

We have chosen to use 1x16000 mah at 24v and 2x8000 mah at 12v in parallel Lithium Polymer batteries for our vehicle. This separates the high current of the motors from the other low current devices. The capacity was chosen after considering the power consumption of all our onboard devices. Our 12v batteries are of 96Wh therefore our bot will run for 65 minutes on maximum load. Our 24v battery has 380Wh therefore our motors will run for 95 minutes on maximum load.

Battery	Name	Power Consumption
12V 8000mah*2 96 Wh	Jetson Tx2	60W
	Sparton AHRS-8P	5W
	ZED F9P	5W
	Misc. 1.Arduino 2.Lights 3.Relay 4.Zigbee	15W
	Total	85W
24V 16000mah*1 380 Wh	Motors	200W
	Velodyne VLP-16	36W
	Total	236W

We are using the kill switch to cut off power to our motor drivers, hence the motors. The power lines of the motor drivers have a heavy-duty relay placed in series. The normally closed and common contacts are used for this series connection. This saves power as the coil is only turned on when the kill switch is pressed. For the wireless e kill, we use the signal given by our XBEE module to switch on another small relay which in turn switches on the main relay coil. All this circuitry is present on our custom PCB.

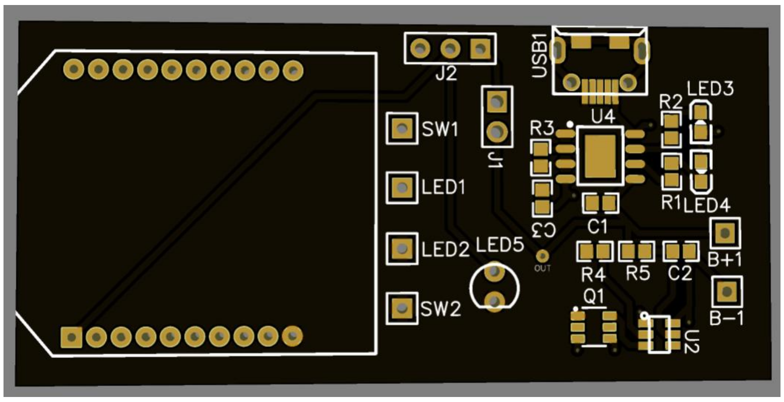

Fig 10: Wireless E-Kill Switch PCB

5.2 Power regulation

We are using 2 distribution boards one for 24v and the other for 12v power. The 24v power distribution is connected to the motors and lidar. The 12v is connected to a buck boost board to stabilize the 12v for the other components. This regulated 12v is then used for powering the remaining components.

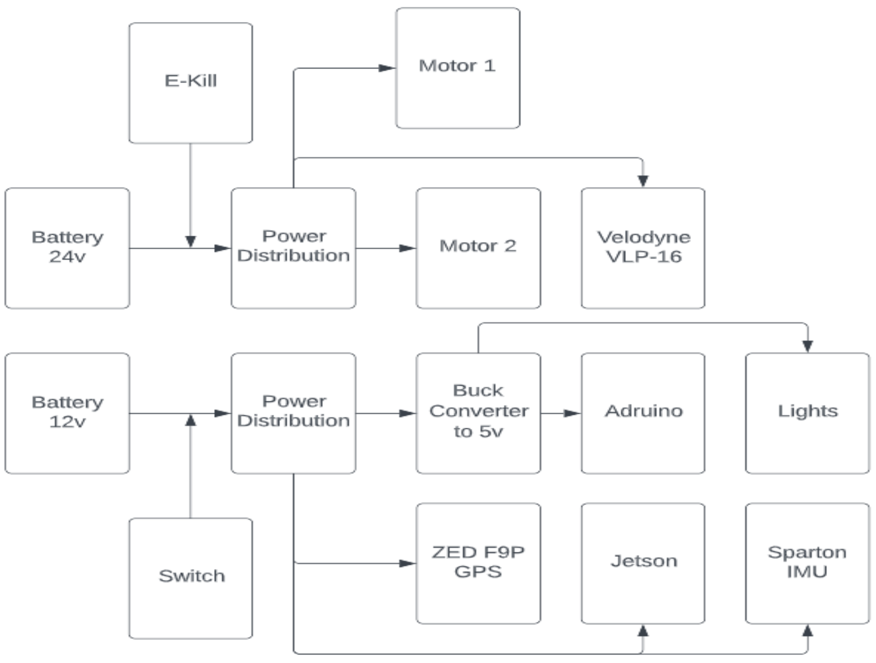

Fig 11: Power regulation Circuit

5.3 Control PCB

For connecting the motor drivers and Jetson, we have designed a PCB based on the ATMEGA2560. This PCB contains all the necessary components and wiring required to run the motors by reading serial commands from the Jetson. It also reads the encoder pulses and publishes the odometry information. This PCB contains a secondary microcontroller for diagnostics such as reading battery voltage, temperature, battery fan speed control, light switching and other small tasks.

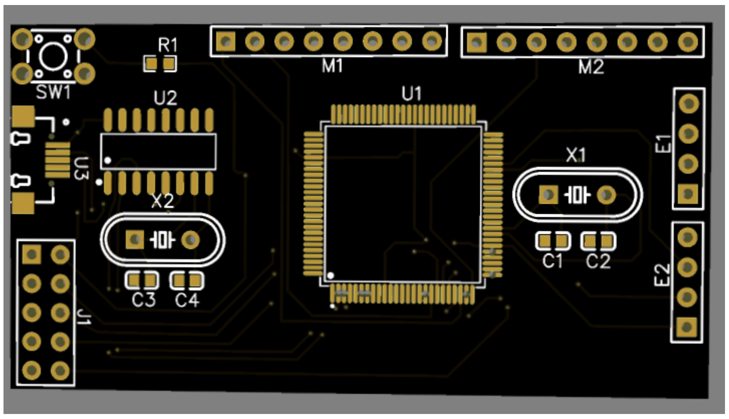

Fig 12: Control PCB

6 Software Design

6.1 ROS

We decided to use ROS in our software pipeline as it solved many of our issues. ROS provides a node-subscribe-publish interface that works more efficiently. Each task can be divided into separate nodes and run individually or together. This makes testing and error handling very easy. For using our Lidar, we are using the ROS VELODYNE
package only for parsing the sensor data which publishes the pointcloud and laserscan data, the actual processing is done by our pipeline. For the IMU, a ROS driver for the sparton AHRS-8P was found on the ROS wiki and publishes the orientation information of the vehicle. Our ROS architecture is shown in the figure below.

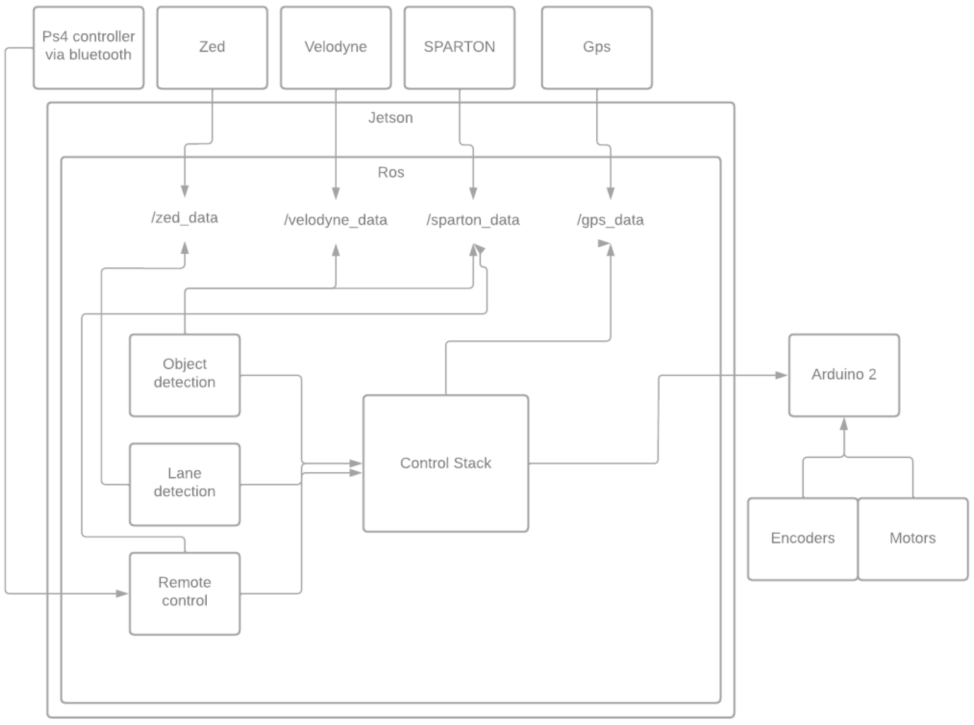

Fig 13: ROS System

6.2 IMU Control

Our first aim was to control the robot with a Bluetooth controller based on the ROS cmd_vel commands. The controller would send a command consisting of 2 values - the linear speed and the steering angle. We have chosen the SPARTON AHRS-8P as our IMU and it constantly publishes the vehicle's orientation and velocities. A PID controller then tries to minimize the error between the command and current state. This also helps Socrates to drive in a straight line as without any correction, due to the difference in the motor rpm's the robot drives in an undesired path.

6.3 Object Detection and Avoidance

For detecting objects in front of Socrates we are using the Velodyne LiDAR. The output of the lidar is the form of a list of distances at each angle. This list can be plotted and worked with by using simple trigonometry as shown in the diagram below. The next step is to separate the points and their angles into 2 categories - objects and not objects.

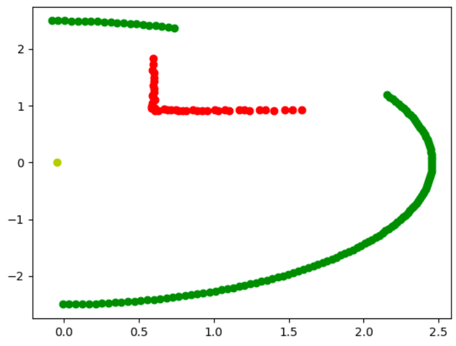

Fig 14: Object Detection

After classification, any angle from the not object angle list has to be chosen and it will cause the robot to avoid the object. An improvement in this algorithm is that when choosing the object avoidance angle, we can choose it in such a way that the angle chosen is closest to the command angle from the joystick/lane detection/GPS navigation pipelines. This allows the robot to avoid objects as well as follow the command of the other pipelines.

6.4 Mapping

By using the encoder odometry and transform information, mapping is performed. We have not used readymade mapping algorithms and instead gone for an approach from scratch. We have created our own mapping tool based on matplotlib. This mapping module logs and plots the lidar data which is color coded - red for object and green for no object. It logs and plots the current GPS position of the bot. It also plots the path that Socrates takes during course navigation. A very basic simulation map that is generated by our tool can be seen below. One lacking feature is that this map cannot currently be reused for other runs so the vehicle does not learn from its previous runs.

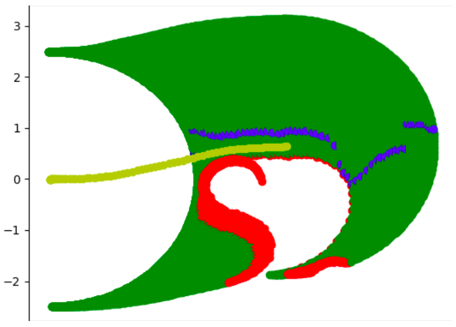

Fig 15: Mapping output of simulation

6.5 GPS Navigation

The GPS and compass readings are constantly monitored by the No man's land detection software module. The current coordinates are compared with the waypoints and if found equal within the range of error then the lane detection pipeline data is ignored. The compass is used for calculating the offset angle to the next waypoint. This data is used to guide the vehicle towards the next waypoint. The object detection data is still being used. When the vehicle reaches the next waypoint, the whole process is repeated. This is done n number of times where n is no of waypoints.

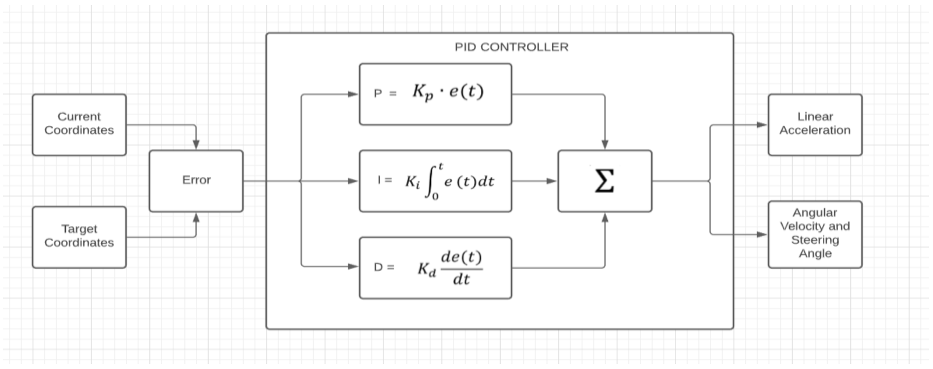

Fig 16: GPS Navigation

6.6 Lane Detection and Following

For detection of lanes, we are using an image processing-based pipeline. We're using a method in which the histogram of the rows and columns of the image is calculated to find out the coordinates of the lanes. A mid-point is then found from these points and then the steering angle is calculated. This angle is sent to the IMU control module which drives the robot at that angle which makes it follow the lanes. This method has been tested on simulation and by adjusting the HSV values, it can also work in different conditions.

7 Failure Modes

We have identified some failure modes in Socrates and have listed them below:

Category	Failure Point	Explanation
Mechanical	Coupler	During a high radius turn, due to frictional forces the coupler comes loose.
Electrical	IMU	Due to magnetic interference the compass points to an incorrect north
	Wiring	Loose connections may cause short circuits
	Battery	Excessive heat may cause the battery to catch fire
Software	Lane detection module	In some conditions, the lane is not clearly visible which causes errors
	Object detection and avoidance module	If the object is very close to the lidar, it is not detected because of the lidar's minimum range

8 Simulations

We created Socrates in Gazebo, a 3D simulator that was essential for the robot's software testing. Socrates's model was loaded into Gazebo, which allows for precise, dynamic interactions inside the simulation. The LiDAR and vision sensors on Socrates were simulated independently using their respective libraries. We created an AutoNav course, to ensure Socrates can navigate itself around the challenges faced. This course included a No-Land, Man's and a custom-designed path with edge cases that would be difficult to get around.

These simulations were extensively used in testing lane and object detection. Our team was able to try out different path planning strategies quickly and effectively. This tool was extremely useful in the COVID-19 pandemic when the team was unable to meet physically.

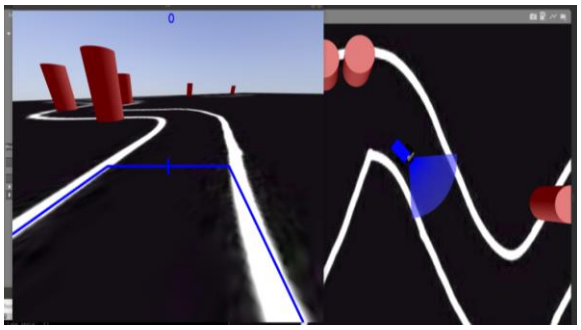
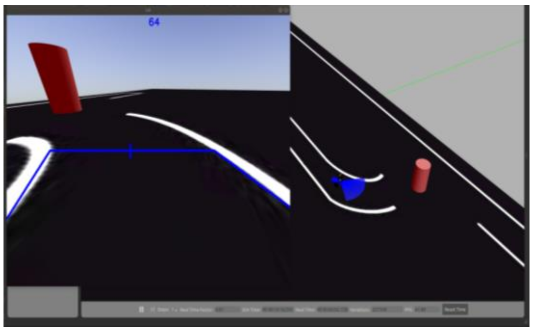

Fig 17 and 18: FOV of AutoNav Course

9 Performance Testing

At the time of writing this report, all our software had been tested thoroughly on Gazebo simulator. In practice, lane detection and following along with GPS navigation was tested on a ground and the vehicle performed well on the track.

10 Initial Performance Assessment

During our initial testing we found that Socrates performs well. It maintains an average speed of 2.5mph with the payload. It can easily climb and descend the 15-degree ramp with the payload with high stability. Its battery life is very close to the calculated run time due to losses.

11 Cost

Item Name	**Cost (INR)**	**Cost to team (INR)**
Jetson	50,000	25,000(Sponsor Discount)
Sparton	1,50,000	0(Sponsored)
Lidar	7,00,000	0(Sponsored)
Wheels	5,000	5,000
Motors	15,000	15,000
ZED	40,000	40,000
Chassis and Misc.	40,000	40,000
Total	10,00,000	1,25,000