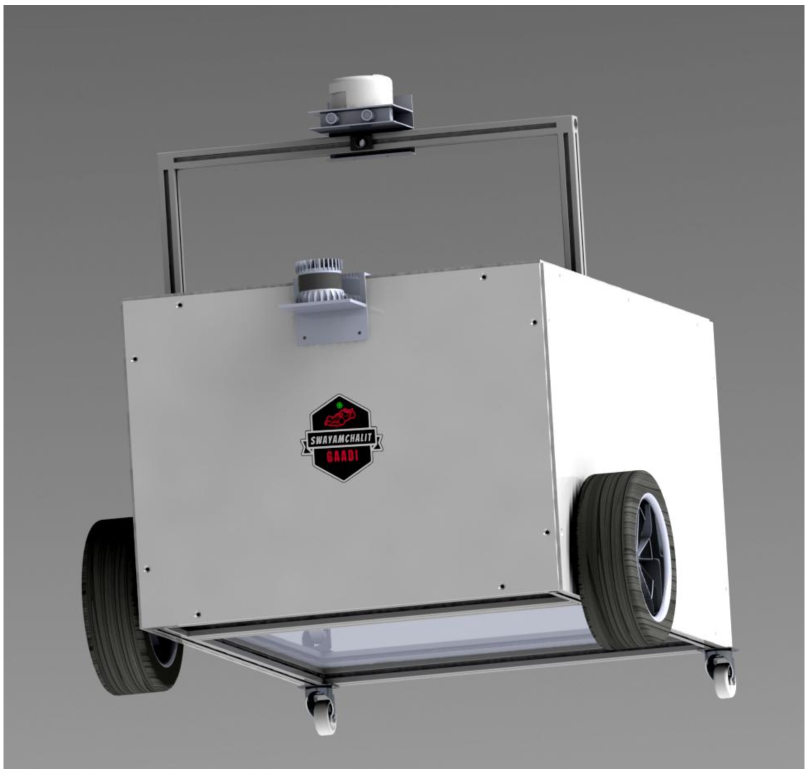
KALINGA INSTITUTE OF INDUSTRIAL TECHNOLOGY

PROJECT SWAYAMCHALIT GAADI

SwaG MARK-1



Team Leader
Anshuman Parhi | 1704500@kiit.ac.in

Team Manager
Anand Golla | 1707102@kiit.ac.in

Software Dept Head
Maitreya Mohapatra | 1705242@kiit.ac.in

Electrical & Electronics Dept Head
Sourav Sahoo | 1707152@kiit.ac.in

Mechanical Dept Head
Mrinmoy Roy | 1702239@kiit.ac.in

TEAM MEMBERS

Mechanical

Mrinmoy Roy	1702239@kiit.ac.in
Ayush Kumar	1704258@kiit.ac.in
Sayan Pal	1704552@kiit.ac.in

Electrical and Electronics

Anand Golla	1707102@kiit.ac.in
Sourav Sahoo	1707152@kiit.ac.in

Software

Mailtreya Mohapatra	1705242@kiit.ac.in
Ishan Gupta	1705316@kiit.ac.in
Anshuman Parhi	1704500@kiit.ac.in
Ronak Sengupta	1704546@kiit.ac.in

Networking and Cyber-security

Sujit Ku. Sahoo	1704689@kiit.ac.in
Dwitindra N Sahoo	1704110@kiit.ac.in
Abhisek Omkar	1929302@kiit.ac.in

Faculty Advisor
Prof.(Dr.) Isham Panigrahi | ipanigrahifet@kiit.ac.in

May 1, 2020

➢ Who are we?

I. About Project Swayamchalit Gaadi (SwaG)

Swayamchalit Gaadi is the Autonomous Ground vehicle team of KiiT Deemed to be University, India. We are a group of highly driven and motivated polymaths who develop, envision, and engineer cutting-edge technology in the field of autonomous mobility. SwaG Mark-1 is the first in the upcoming series of Intelligent Ground Vehicles entirely designed and built according to the requirements of IGVC 2020. This model embeds superb navigation, path-planning and Image Processing techniques to accomplish the tasks. The sensor fusion of this model comprises of a 3d Lidar, Stereo-camera and Wide-angle camera for visual odometry and SLAM. GPS with RTK corrections have been used for precise navigation of the way-points. As of the computing is concerned, we have avoided high power processing units for an extended run-time and cost-efficiency, instead we have used a Kubernetes cluster of Nvidia Jetson Nano and Raspberry Pi for task-specific operations.

II. Organizational Structure

Team SwaG consists of four departments, 3 sub-teams, a Team Manager and a Team Leader. The departments correspond to the major disciplines of IGVC: mechanical, electrical and electronics, networking / cybersecurity and software. Each department has a department head, responsible for coordinating the details of the team's technical work and introducing new members to the team. The sub-teams: Management, Tech, and Marketing were responsible for the proper functioning of the team and maintain the work-flow.
The team's management sub-team acts as watchdog of all team activities, coordinates department specific tasks and lays out the timeline whereas marketing sub-team is responsible for team funding, sponsorship operations and branding.The tech sub-team however forms the core of the team and is responsible for the development of the model. Over the course of the 2019-2020 year, the team has put in over 4000 hours researching, designing and manufacturing of SwaG Mark-1. To satisfy the interdisciplinary aspects of the project some members of the team had to multitask to maintain proper balance.

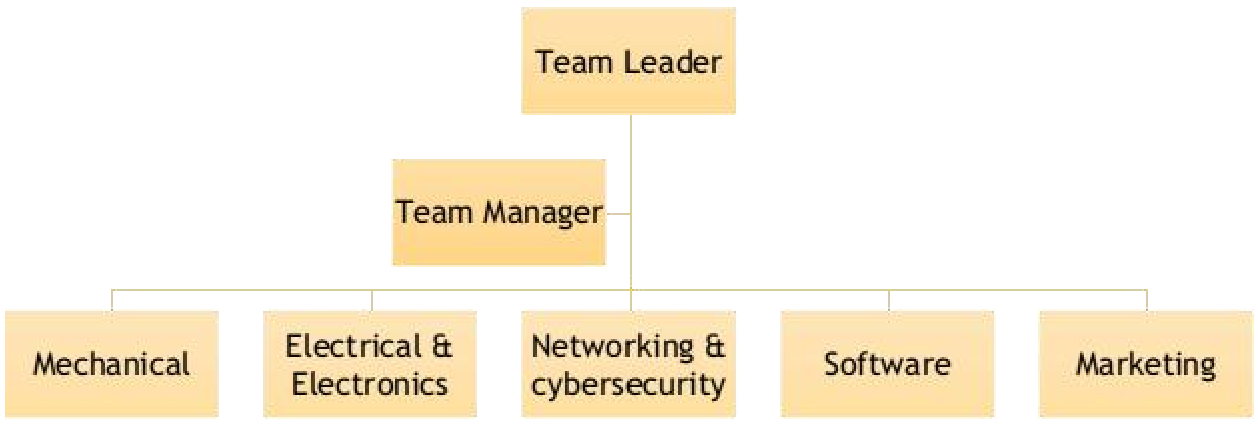


Figure 1. Organizational chart

Following table exhibits of profile of our team members-

Department	Name of the Member	Year of Study	Branch of Study
Team Leader and ROS team member-I	Anshuman Parhi	3rd year undergrad	Electronics and Telecommunication engineering
Team Manager and core Electronics member	Anand Golla	3rd year undergrad	Electrical and Electronics engineering
Mechanical Head	Mrinmoy Roy	3rd year undergrad	Automobile engineering
Core Mechanical member-I	Ayush Kumar	3rd year undergrad	Electronics and Telecommunication engineering
Core Mechanical member-II	Sayan Pal	3rd year undergrad	Electronics and Telecommunication engineering
Electrical and Electronics Head	Sourav Sahoo	3rd year undergrad	Electrical and Electronics engineering
Software Head	Maitreya Mohapatra	3rd year undergrad	Computer Science engineering
Core Software developer	Ishan Gupta	3rd year undergrad	Computer Science engineering
ROS Team member-II	Ronak Sengupta	3rd year undergrad	Electronics and Telecommunication engineering
Core Networking member and Marketing Head	Dwitindra Nath Sahoo	3rd year undergrad	Electronics and Telecommunication engineering
Cybersecurity Head	Sujit Kumar Sahoo	3rd year undergrad	Electronics and Telecommunication engineering
Core cybersecurity member	Abhisek Omkar Prasad	Fresher !st year undergrad	Computer Science engineering

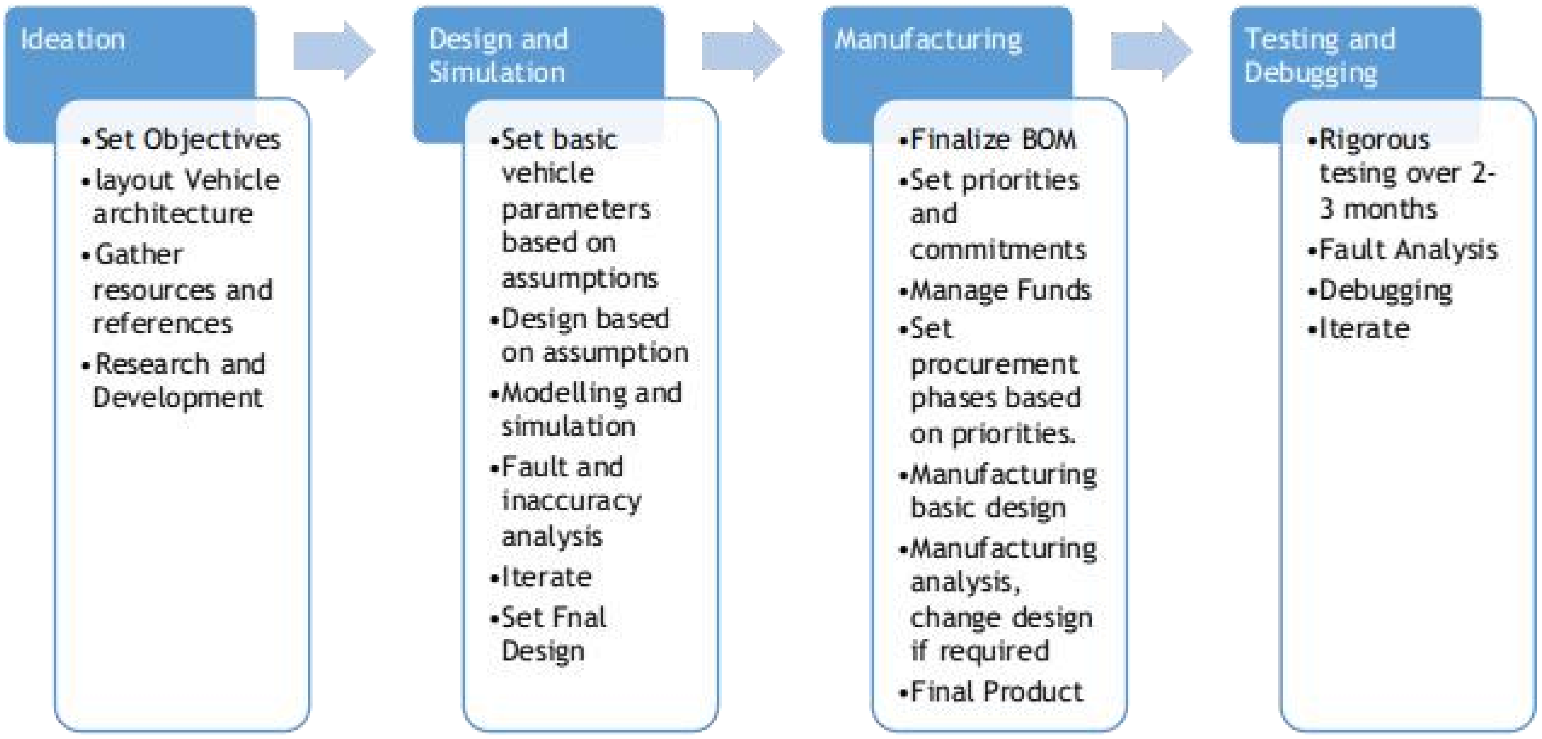
Table i- Team Member profile

Manangement sub-eam	Core Tech sub-team	Marketing sub-team
•Anshuman Parhi	•Mrinmoy Roy	•Dwitindra Nath Sahoo
•Anand Golla	•Ayush Kumar	•Anshuman Parhi
•Sujit Kumar Sahoo	•Sayan Pal	•Anand Golla
	•Sourav Sahoo	
	•Anand Golla	
	•Maitreya Mohapatra	
	•Ishan Gupta	
	•Anshuman Parhi	
	•Ronak Sengupta	
	•Sujit Kumar Sahoo	
	•Abhisek Omkar Prasad	

Table ii- Sub-team Division

III. Work Process

Our work process is primarily divided into 4 simple processes- Ideation, Design and Simulation, Manufacturing, and Testing and Debugging. These 4 processes occur in either full iteration or partial iteration until the desired objective is achieved.



Ideation
•Set Objectives
•layout Vehicle architecture
•Gather resources and references
•Research and Development
Design and Simulation
•Set basic vehicle parameters based on assumptions
•Design based on assumption
•Modelling and simulation
•Fault and inaccuracy analysis
•Iterate
•Set Fnal Design
Manufacturing
•Finalize BOM
•Set priorities and commitments
•Manage Funds
•Set procurement phases based on priorities.
•Manufacturing basic design
•Manufacturing analysis, change design if required
•Final Product
Testing and Debugging
•Rigorous tesing over 2-3 months
•Fault Analysis
•Debugging
•Iterate

Figure 2. Iterative Work flow

Starting completely from scratch, our work process was highly dependant on open source resources, so it began all with hours of online research , studying various design reports, chalking out rough vehicle structure and suite and again redesigning them. We decided to keep the Mechanical structure simple yet reliable enough to carry the whole weight along with the payload. The CAD designing was iterated almost 4 times before reaching the final model. The same was repeated for electronics, where we went from a high power gaming PC in the beginning for computation to a kubernetes cluster of Nvidia Jetson Nano , thereby achieving a very small form factor for our IGV. The software however was consistent through out the whole process, we started from testing very basic codes on smaller RC prototypes to implementation of ROS framework in the vehicle.

IV. Mechanical Suite

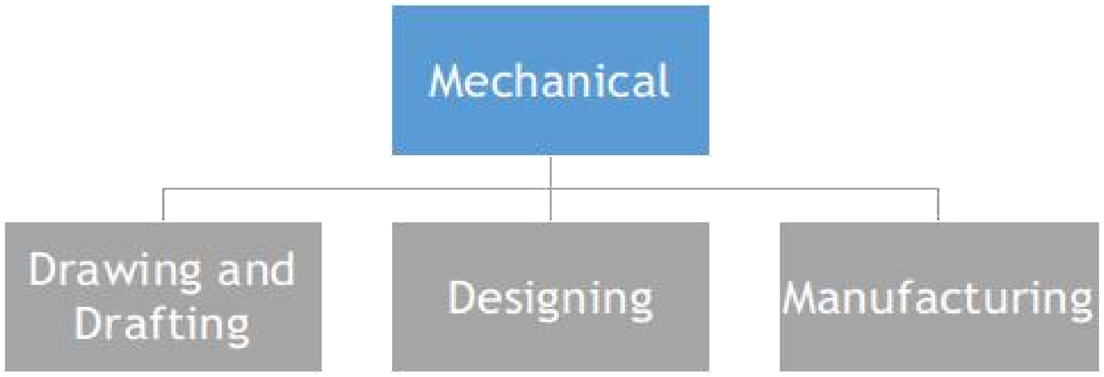


Figure 3. Mechanical Suite division

Design Objective

The objective of this project is to design a vehicle that can analyse and define its path on its own with constricted dimensions (height, length, breadth), according to IGVC guidelines. The vehicle should also satisfy optimum stability criteria.

Design Approach

The team considered the blueprint of the vehicle as a system design project and system engineering approach was followed to complete the project on time. The detail approach is described below-

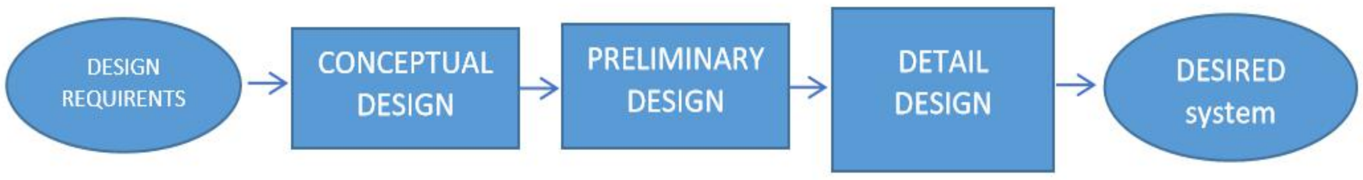


FIGURE 1: Flowchart for illustration of development phase

Figure 4. Design Approach

Conceptual Design

The team has had many sessions to allow its members to come up with new ingenious ideas and theories to finally evolve a design that has minimum fault. To provide the design of an unusual yet achievable vehicle with best performance, theoretical research work was also done by its members, individually as well as collectively.

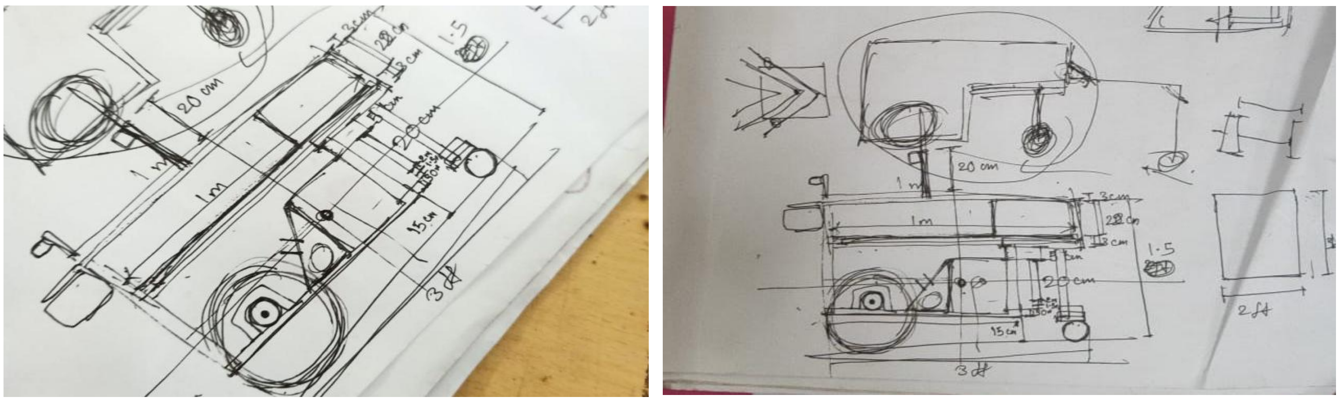


Figure 5. Rough Vehicle Sketc

Preliminary Design

The outputs of conceptual design serves as the input for preliminary design phase. Purpose of preliminary design was to perform a sizing analysis like, motor torque estimation, wheel type and dimension and material requirements, however the outcomes of various calculation and parameters determined were not final and would be subjected to altercations at later stages. The design process was divided into various sections for greater overall efficiency pertaining to inputs available.

Torque Calculation For Motor

The team has calculated the required torque according to the following research paper and minor changes has been done as per requirement.

- Gross Vehicle Weight : 40 kg
- Radius of Wheel : 8 inch
- Desired top speed : 1.5 ft/sec
- Desired acceleration time : 1 sec
- Maximum inclined angle : 2 degree

Now, **Total Tractive Effort(TTE)** Calculation -

$$\boxed{TTE = RR + GR + FA}$$

Where,

TTE - Total tractive Effort
RR - Rolling Resistance
GR - Grade Resistance
FA - Acceleration Force

- **Determine the rolling resistance**

$$\boxed{RR = GVW + Crr}$$

Where,

GVW - Gross Vehicle Weight
Crr - Coef. Of Rolling resistance
RR = 40 x 0.75(Coef. Of Rolling resistance for soft Grass**)**
= 30 kg.

- **Determine Grade resistance**

$$\boxed{GR = GVW \times \sin\theta}$$

Θ = Maximum inclined angle (20°)
GR = 40 x Sin(20°)
= 13.68 kg.

- **Determine acceleration Force**

$$\boxed{FA = {}^{(GVW\,'\,V\max)}\!\big/_{(g\,'\,t)}}$$

Vmax = Max. Speed
t = time required to achieve max. Speed
g = Acceleration of gravity
FA = (40 x 1.65 km/hr) / (35.28(g in km/hr) x 0.000277778)
= 1.86 kg.

- **Total Tractive Force**

TTE = RR + GR + FA

= 30 kg + 13.68 kg + 1.86 kg
= 45.54 kg

- **Determine Wheel Motor torque**

Tw = TTE x Rw x RF

Where,
Rw = wheel Radius
RF = Resistance Factor
Tw = 45.54 kg x 8 inch x 1.1
= 400.752 kg-inch (1 inch = 6.35 kg)

- **Peak RPM**

$$TopSpeed\,(km/hr)´\ {}^{5}\!/_{18}´\ 60 \Big/ {}_{2´\ 3.14´\ r}$$

r = Radius of tyre
Top Speed = 1.65 km/hr
(1.65x(5/18)x60) /(2x3.14x8)
= 0.55 RPM

Detailed Design

The vehicle is front wheel drive with two caster wheels in back side of the vehicle.

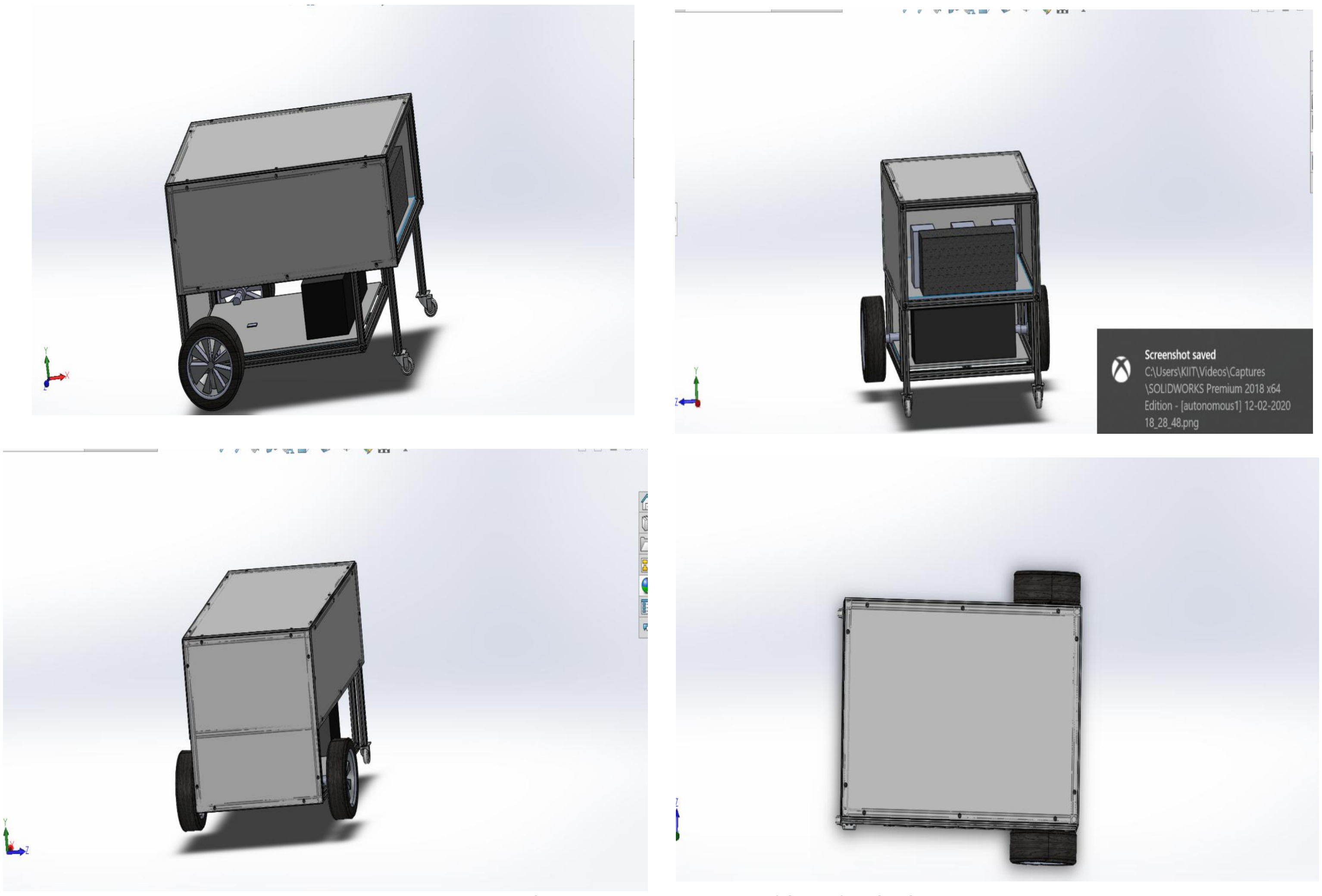

Screenshot saved
C:\Users\KIIT\Videos\Captures
\SOLIDWORKS Premium 2018 x64
Edition - [autonomous1] 12-02-2020
18_28_48.png

Figure 6. Side/Top/Rear/Front Profiles of Vehicle

After some critical thinking and analyzing, team decided to change the vehicle model to make it more compact and space efficient.

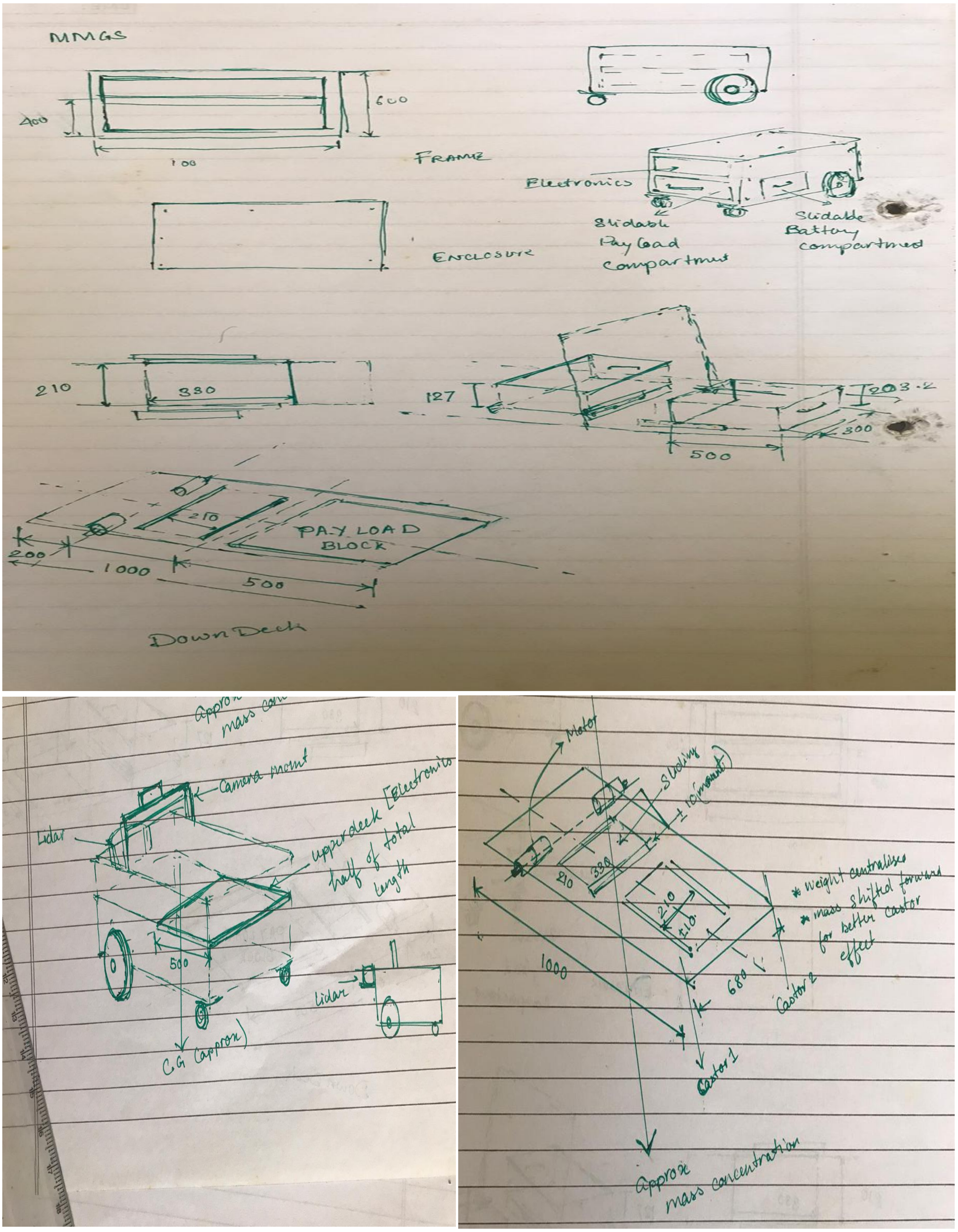


Figure 7. Re-planning for compact vehicle structure

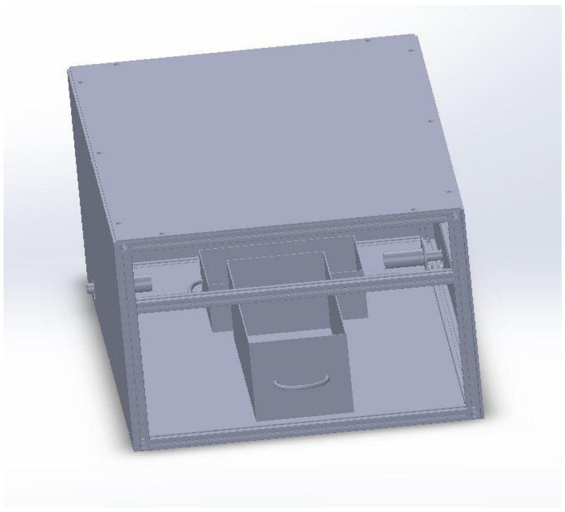
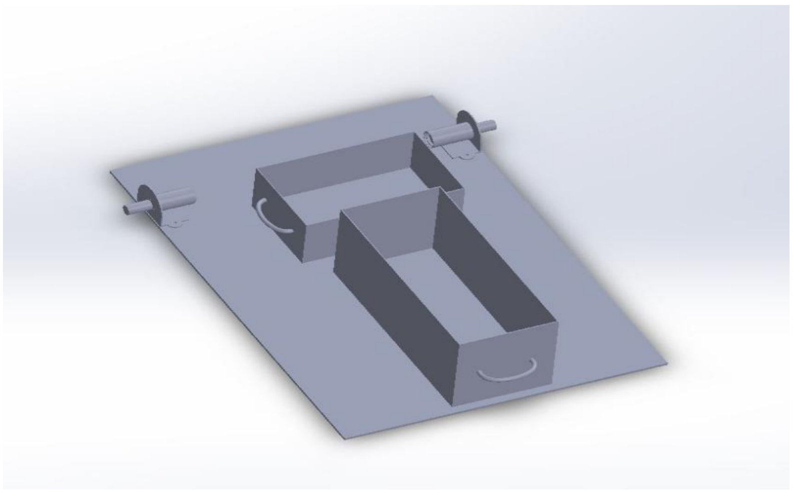
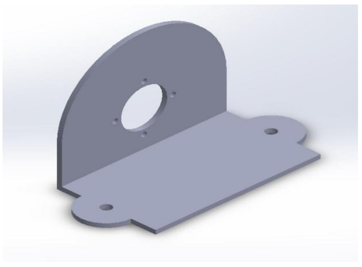
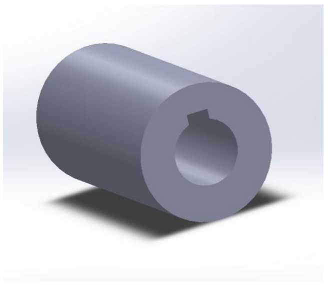



Figure 8. This design is more efficient, compact and stable than previous design.

Motor and Motor Mount

Two maxon motors have been used to make the vehicle more dynamic and to be able to run on grassy surface.The team has designed the customized motor coupling which act as a connector between tyre and motor and along with Motor mount has been designed. The vehicle is front wheel drive with two tyres having 8-inch diameter in the front part of the vehicle and two caster wheel in the back side of the vehicle.

Weather Proofing

To overcome the weather proofing problem the main housing unit is lined with silicone based sealant on the edges leaving no area inside exposed to any kind of external weather threats. Apart from this the vehicle is covered with HDPE (High Density Poly Ethylene) sheets to make sure that the sensors and connections which are exposed are covered.

V. Electrical and Electronics Suite

Battery Pack Designing

Estimated run time = 1 Hours.
Total Load of the electrical system = 45A
Required Voltage of the system = 24V
We have used 18650 Li-ion batteries with each cell having a nominal voltage of 4.2V and capacity of 2700mAh or 2.7Ah.
In order to fulfil the voltage requirement no of cells in series is calculated as below-

Total voltage=24V
Voltage of each cell=4.2V

No of cells in series = $\frac{Total\ voltage\ required}{Voltage\ of\ each\ cell}$

= $\frac{24V}{4.2V}$

=5.71 = 6 Cells

Therefore we need 6 No of cells in series to fulfil our voltage requirement.
In order to fulfil the amperage requirement no of cells in parallel is calculated as below-

Total Amperage required =47 Ah
Assuming a safety factor of about 5Ah
New Amperage = 52Ah
Discharge capacity of each cell = 2.7Ah

No of cells in parallel = $\frac{Total\ capacity\ required}{capacity\ of\ each\ cell}$

= $\frac{52Ah}{2.7Ah}$

= 19.25 = 20 cells

Therefore we need 20 No of cells in parallel to fulfil our Amperage requirement.
Total No of cells required = 6 * 20= 120 Cells
Thus, the required battery pack consists of 6 series and 20 Parallel cells i.e. 6s20p.

Electrical Load Calculation

We have used the following components:-

Equipments	Quantity	Amperage	Net Amperage
Motors	2	10 amps	20 amps
LIDAR	1	1.5amps	1.5 Amps
GPS	1	1 Amps	0.1 Amps
Camera	2	1 Amps	2 Amps
Jetson	2	4 Amps	8 Amps
Raspberry Pi	2	3 Amps	6 Amps
Arduino	2	2 Amps	4 Amps
Auxiliaries		5 Amps	5 Amps
		Total Load	46.6 Amps

Table iii. PDS

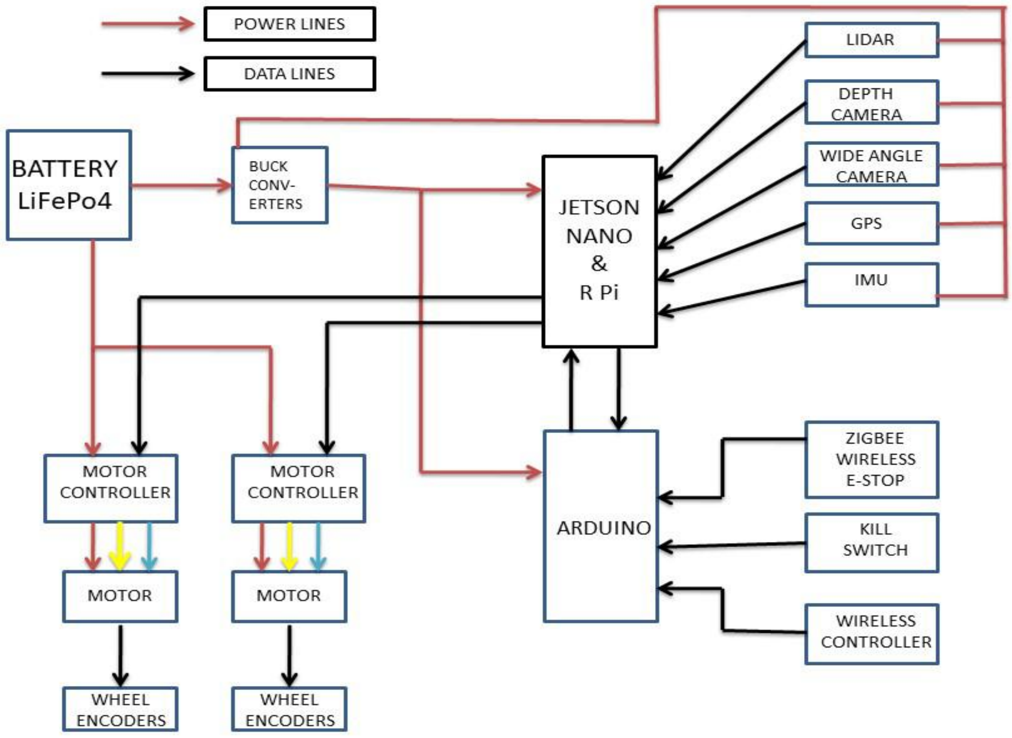


Figure 10. Electrical system block diagram

Battery Pack Manufacturing

For the first time, we have made a 24V 54Ah in- house Battery Pack and insured that no HV runs in any parts of the vehicle unless master switch is turned ON.

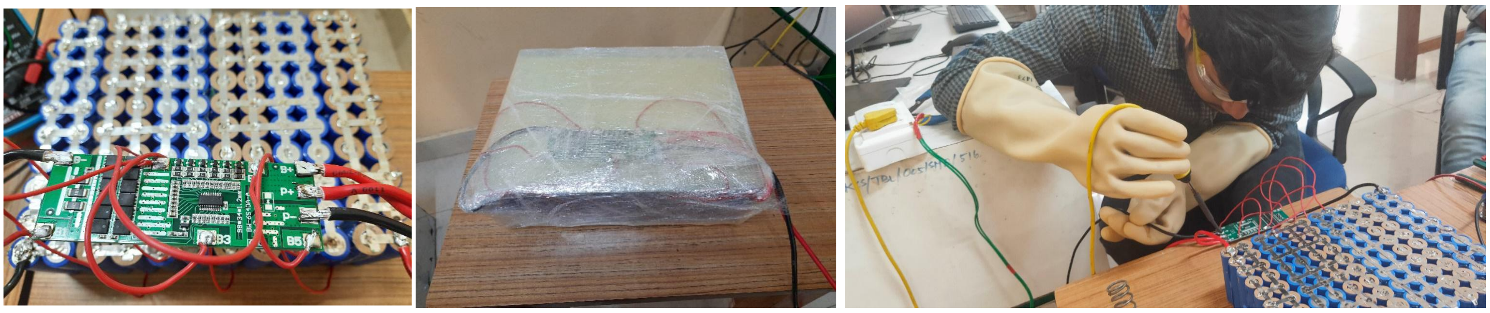


Figure 11. In-house battery pack manufacturing

In-House PCB Design

We designed our circuits and printed our own PCBs for auxiliary connections of motor controller and other circuits wherever needed with the help of Autodesk Eagle.

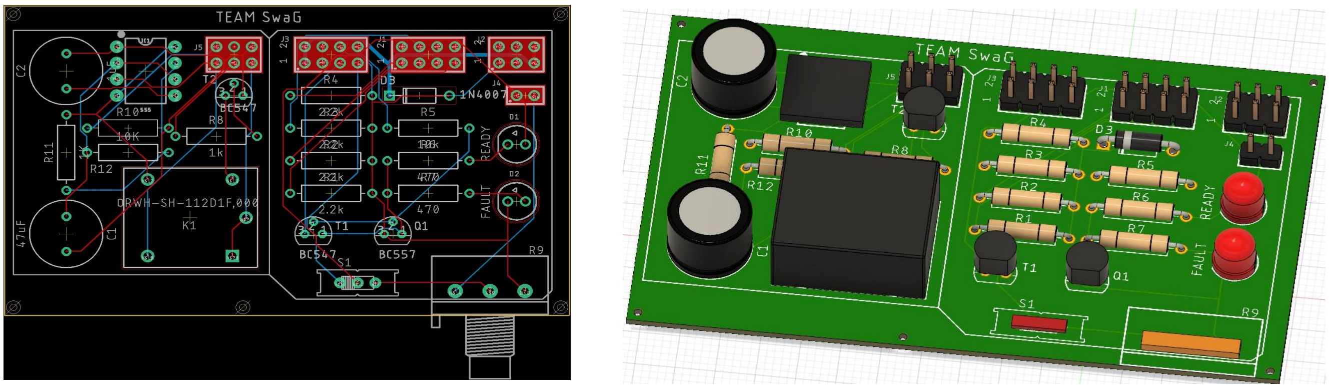


Figure 12. Auxiliary circuit for Motor controller

Fault Response

1. Fuse is used in some places to ensure no current higher than the rated flows through the circuit.
2. When the algorithm detects any error with the functioning it throws an error signal to the Arduino which in turn activates a relay to switch off the power to motors and other parts which are expected to get damaged due to false instructions. This relay acts as a kill switch for the parts which are most vulnerable to damage. This way our sensitive parts are safe and for logging the error data the system is still powered on for manual reset.
3. Shut down circuit with Kill Switch is laid out for manual stop in case of any failure.
4. Wireless E-Stop with the help of ZIGBEE is also interfaced with the Arduino and included in the shutdown circuit.

VI. Software Suite

Overview

The software system for SWAG 1.0 is built on Robot Operating System (ROS). The system is divided into 2 parts namely perception and planning. The perception module interprets the incoming data from various sensors. It is responsible for localizing obstacles, lanes, potholes and also Identification of road signs. The planning module accepts the interpreted data for creating a cost map of the environment which distinguishes between feasible and obstructed regions for the robot to travel. Path and trajectory planning algorithms then generate the path to be taken by the robot to reach the destinations without any fatalities or collisions.

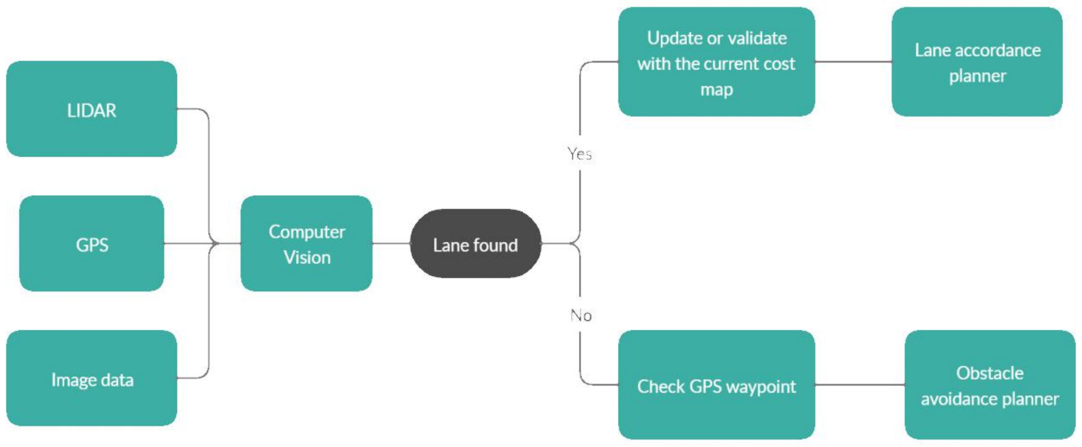


Figure 13. Basic system layout from software perspective

ROS Overview

ROS (Robot Operating System) is released by OSRF (Open Source Robotics Foundation) as an open source project. It is a software platform which provides a framework of communication layer and builds system for robotic software. ROS runs mainly on Linux. Motivation of ROS is to support software reuse and to build robotic systems with software components in robotic research and development.

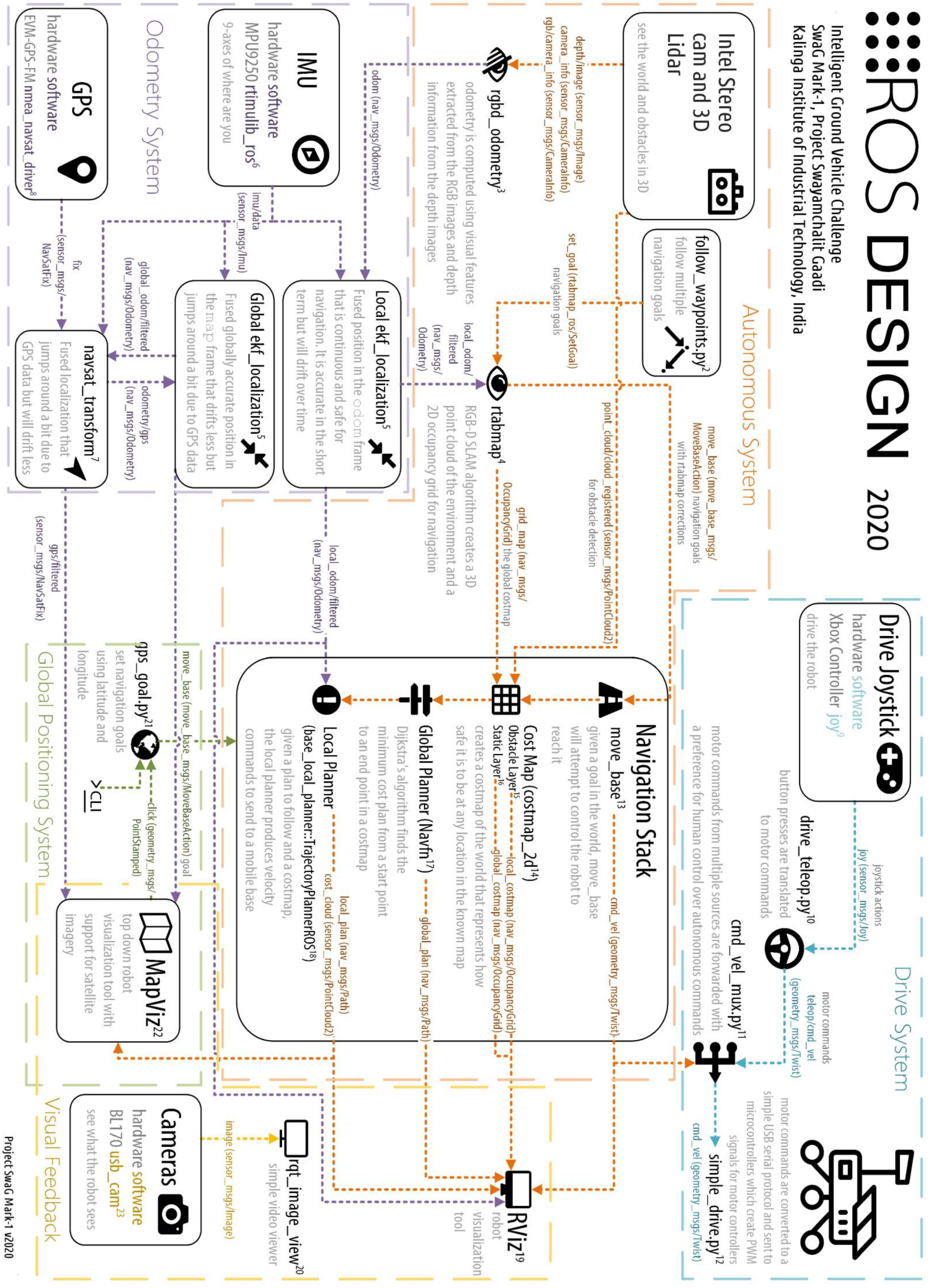

ROS DESIGN 2020
Intelligent Ground Vehicle Challenge
SwaG Mark-1, Project Swayamchalit Gaadi
Kalinga Institute of Industrial Technology, India
Autonomous System
Intel Stereo cam and 3D Lidar
see the world and obstacles in 3D
depth/image (sensor_msgs/Image)
camera_info (sensor_msgs/CameraInfo)
rgb/camera_info (sensor_msgs/CameraInfo)
rgbd_odometry3
odometry is computed using visual features extracted from the RGB images and depth information from the depth images
odom (nav_msgs/Odometry)
follow_waypoints.py2
follow multiple navigation goals
set_goal (rtabmap_ros/SetGoal) navigation goals
rtabmap4
RGB-D SLAM algorithm creates a 3D point cloud of the environment and a 2D occupancy grid for navigation
local_odom/filtered (nav_msgs/Odometry)
move_base (move_base_msgs/MoveBaseAction) navigation goals with rtabmap corrections
point_cloud/cloud_registered (sensor_msgs/PointCloud2) for obstacle detection
grid_map (nav_msgs/OccupancyGrid) the global costmap
Odometry System
IMU
hardware software
MPU9250 rtimulib_ros6
9-axes of where are you
imu/data (sensor_msgs/Imu)
GPS
hardware software
EVM-GPS-FM nmea_navsat_driver8
fix (sensor_msgs/NavSatFix)
Local ekf_localization5
Fused position in the odom frame that is continuous and safe for navigation. It is accurate in the short term but will drift over time
Global ekf_localization5
Fused globally accurate position in the map frame that drifts less but jumps around a bit due to GPS data
global_odom/filtered (nav_msgs/Odometry)
odometry/gps (nav_msgs/Odometry)
navsat_transform7
Fused localization that jumps around a bit due to GPS data but will drift less
local_odom/filtered (nav_msgs/Odometry)
gps/filtered (sensor_msgs/NavSatFix)
Drive System
Drive Joystick
hardware software
Xbox Controller joy9
drive the robot
joystick actions joy (sensor_msgs/Joy)
drive_teleop.py10
button presses are translated to motor commands
motor commands teleop/cmd_vel (geometry_msgs/Twist)
cmd_vel_mux.py11
motor commands from multiple sources are forwarded with a preference for human control over autonomous commands
simple_drive.py12
motor commands are converted to a simple USB serial protocol and sent to microcontrollers which create PWM signals for motor controllers
cmd_vel (geometry_msgs/Twist)
Navigation Stack
move_base13
given a goal in the world, move_base will attempt to control the robot to reach it
cmd_vel (geometry_msgs/Twist)
Cost Map (costmap_2d14)
Obstacle Layer15
Static Layer16
creates a costmap of the world that represents how safe it is to be at any location in the known map
local_costmap (nav_msgs/OccupancyGrid)
global_costmap (nav_msgs/OccupancyGrid)
Global Planner (Navfn17)
Dijkstra's algorithm finds the minimum cost plan from a start point to an end point in a costmap
global_plan (nav_msgs/Path)
Local Planner
(base_local_planner::TrajectoryPlannerROS18)
given a plan to follow and a costmap, the local planner produces velocity commands to send to a mobile base
local_plan (nav_msgs/Path)
cost_cloud (sensor_msgs/PointCloud2)
Global Positioning System
gps_goal.py21
set navigation goals using latitude and longitude
move_base (move_base_msgs/MoveBaseAction) goal
click (geometry_msgs/PointStamped)
>CLI
MapViz22
top down robot visualization tool with support for satellite imagery
RViz19
robot visualization tool
rqt_image_view20
simple video viewer
image (sensor_msgs/Image)
Cameras
hardware software
BL170 usb_cam23
see what the robot sees
Visual Feedback
Project SwaG Mark-1 v2020

Figure 14. Detailed Software Architecture

ROS Communication Model

In ROS development, a robot system is designed using a set of components called “node” and its communication channel called “topic”. A robot developer can make a robot software system by using components chosen from a number of distributed packages and build software by connecting them. The developer can also make a node when a new function is needed. Communication model of ROS is based on Publish-Subscribe messaging. Publish-Subscribe messaging is an asynchronous messaging model in which ROS nodes communicate through a topic with each other. The biggest advantage of Publish-Subscribe messaging is a dynamic network configuration. Since the ROS nodes are bound loosely, it is able to add new ROS nodes to the system easily.
There are two roles in ROS nodes: publisher and subscriber. A publisher node publishes a message to a topic. Topic is a classification of message as well as a name of the communication path. A topic holds a sequence of messages. And any subscriber node, which has subscribed to the topic in advance, can receive the messages. Publisher nodes do not know about subscriber nodes. In other words, ROS nodes do not have information on their communication partners. Therefore, ROS system is able to be configured dynamically, which can be said very suitable for rapid system prototyping.

ROS Environment

Our Melodic ROS software built on Ubuntu 16.04 had five main systems:

- the drive system
- the autonomous system
- the global positioning system
- the visual feedback system
- The odometry system

Drive System

The drive software created by Team SwaG is a simple 2 wheel drive because it does not produce wheel odometry or transforms. That is left to the autonomous system via SLAM and is more robust in slippery grassy environments. The simple drive package controls the velocity of 2 maxon ec-i motors with PWM pulses using an maxon motor controller dedicated to driving. The motors are D.C. motors where the voltage on its terminals is given by the duty-cycle of the PWM signal.We have published a new ROS package, simple drive.It proved simple and effective in desert conditions. The package is simple in the sense that it does not publish TF odometry from wheel encoders because wheels slip very substantially on slippery grass. The package implements skid steering joystick teleo-peration with two drive speeds, dedicated left and right thumb-sticks control left and right wheel speeds, a cmd vel multiplexer to support a coexisting autonomous drive system. For the sake of simplicity, this package does not do the following: TF publishing of transforms, wheel odometry publishing,PID control loop, no URDF, or integration with ros control. Though all of these simplifications are normal best practices in sophisticated robots. The simple drive package gives ROS users the ability to advance their robot more quickly and hopefully to find more time to implement best practices.

This package is divided into four parts: drive teleop ROS node, cmd vel mux, ROS node, simple drive ROS node.

Autonomous System

Team SwaG's autonomous system consists of the Intel D345 stereo camera , Ouster OS1-16 3D Lidar and the RTAB-Map ROS package for SLAM mapping. The authors have also contributed gps goal and follow way points ROS packages for added goal setting convenience. The Intel stereo camera , Ouster 3D lidar and ROS wrapper software perform excellently. With the Intel camera and ouster lidar Team SwaG was able to avoid obstacles such as cones and potholes on the auto-nav course . However, the IGV could not move quickly, no faster than slow-moderate human walking speed, because the performance of our on board computer, a Nvidia Jetson Nano, was fully utilized. The Intel camera and Ouster Lidar combined with RTAB-Map for SLAM localization and mapping will work reasonably robustly even in the grassy course where the ground's feature complexity is low and even with a significant amount of shaking on the pole to which our Intel camera was attached.

Navigation System

The navigation stack used by Team SwaG follows the commonly used development patterns of the ROS navigation stack. Our setting choices were inspired by RTAB-Map's tutorial. The new gps goal package uses the WGS84 ellipsoid38 and geographiclib39python library to calculate the surface distance between GPS points. WGS84 is the standard coordinate system for GPS and thus the packages configures GeographicLib to use it because it is important for calculating the correct distance between GPS points.The GPS goal can be set using a geometry msgs/PoseStamped or sensor msgs/NavSatFix message. The robot's desired yaw, pitch, and roll can beset in a PoseStamped message but when using a NavSatFix they will always be set to 0   degrees. The goal is calculated in a ROS coordinate frame by comparing the goal GPS location to a known GPS location at the origin (0,0) of a ROS frame given by the local xy frame ROS parameter which is typically set to 'world' but can be any ROS frame.

Odometry System

At the IGVC competition teams are surrounded by slippery grass terrain. As a result of wheel slippage on grass, Team SwaG didn't use wheel odometry. Instead we focused on fusing IMU and visual odometry into   a more reliable position and orientation. We used the ekf localization ROS nodes of the odometry system. We also relied on odemetry from the rgbd odometry ROS node only and this worked well for our autonomous system based on the RTAB-Map package.

Visual Feedback

To assist in teleoperation of the robot, sensors and navigation plans were visualized in rviz for local information such as point clouds and in mapviz for a global context that includes satellite imagery. The   visual feedback software and joystick software was executed remotely on a laptop in the control station.   The rest of software was executed on a Jetson Nano inside the rover.

Perception System

This is the most crucial segment of out software suite. A stereo Intel Real sense d435 is used for lane detection. This particular model was chosen for its relatively wide 87°±3° x 58°±1° x 95°±3° field of view, depth perception ability. The camera is mounted approximately 1.2 meters off the ground at a 20-degree downward angle.

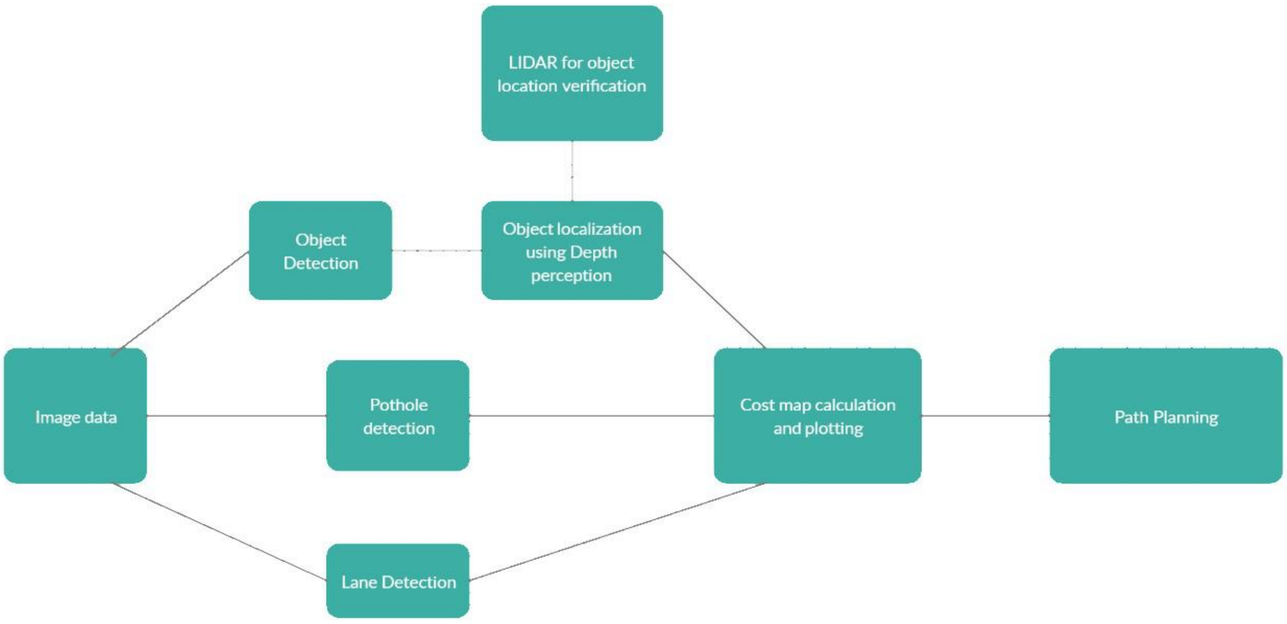

LIDAR for object location verification
Object Detection
Object localization using Depth perception
Image data
Pothole detection
Cost map calculation and plotting
Path Planning
Lane Detection

Figure 15. Perception system block diagram

Lane Detection

Determining Lane Points

The detection of lane lines is done using a modified version of LaneNet. The lane detection is assumed to be a binary segmentation problem which classifies each pixel of input image as point of interest or background. The network is trained to output a binary segmentation map, indicating which pixels belong to a lane and which not. To construct the ground-truth segmentation map, we connect all ground-truth lane points together, forming a connected line per lane.

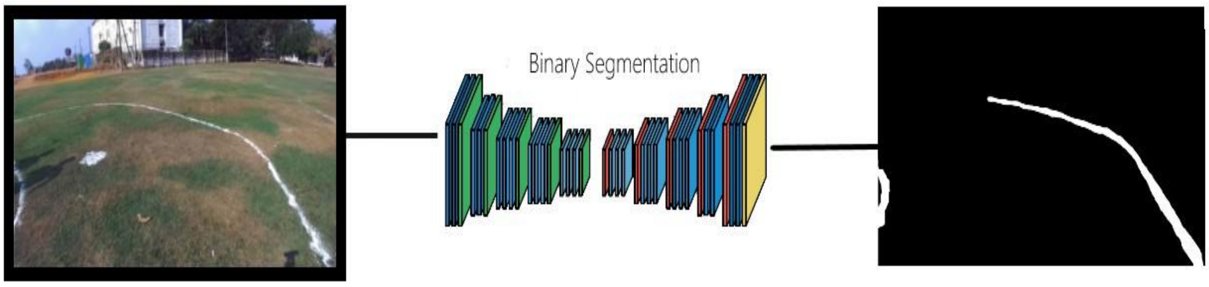


Figure 16. Lane Detection

Localization of lane Points

From the segmented binary mask, we pick out lane points to help plot the lane. This requires accurate localization of the lane points with respect to the robot as well as proper localization and orientation knowledge of the robot itself. We use Intel Real sense, to accurately make a depth map. This is used to segregate the location points and provide the ROS cost map with usable data.

Pothole Detection

The pothole appears as a circle in the inverse - perspective transformed view. We exploit the geometric constraint that the ratio of the perimeter and the square root of the area is constant for circles.

$$\frac{2\ddot{Y}r}{\sqrt{\ddot{Y}r^{2}}}=2\sqrt{\ddot{Y}}$$

Object Detection

Object Detection using YOLO

Object detection is an important part for planning the proper trajectory for the robot. We use an modified version of Mask R-CNN to detect the required objects i.e. cones, signboards and waypoint from the input image. Mask R-CNN is basically an extension of Faster R-CNN. Faster R-CNN is widely used for object detection tasks. For a given image, it returns the class label and bounding box coordinates for each object in the image This data is further processed to make appropriate decisions regarding the path planning of the robot. Similar to the Convnet that we use in Faster R-CNN to extract feature maps from the image, we use the ResNet 101 architecture to extract features from the images in Mask R-CNN. So, the first step is to take an image and extract features using the ResNet 101 architecture. These features act as an input for the next layer.we take the feature maps obtained in the previous step and apply a region proposal network (RPM). This basically predicts if an object is present in that region (or not). In this step, we get those regions or feature maps which the model predicts contain some object.

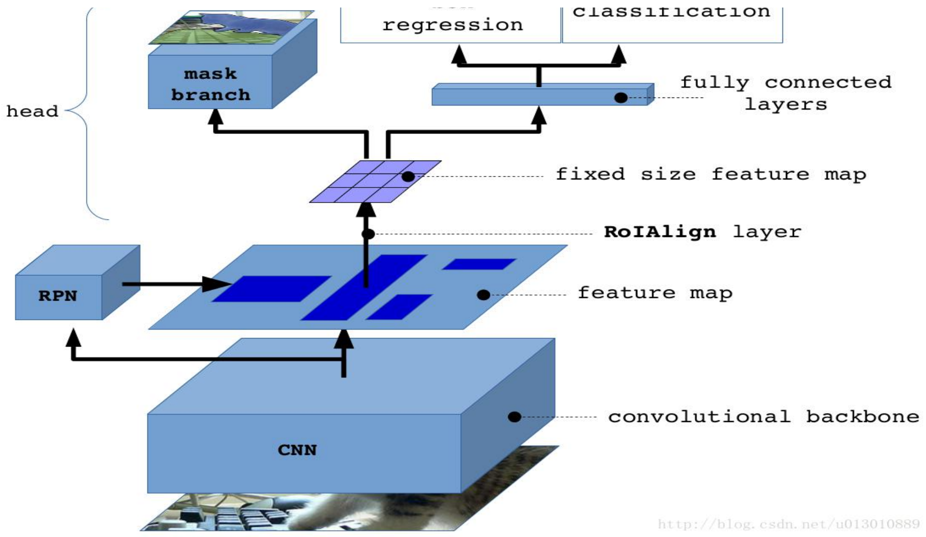


Figure 17. Object detection block-set

Region of Interest

The regions obtained from the RPN might be of different shapes. Hence, we apply a pooling layer and convert all the regions to the same shape. Next, these regions are passed through a fully connected network so that the class label and bounding boxes are predicted. Mask R-CNN also generates the segmentation mask. For that, we first compute the region of interest so that the computation time can be reduced. For all the predicted regions, we compute the Intersection over Union (IoU) with the ground truth boxes. We can computer IoU like this:

IoU = Area of the intersection / Area of the union

Once we have the RoIs based on the IoU values, we can add a mask branch to the existing architecture. This returns the segmentation mask for each region that contains an object. It returns a mask of size 28 X 28 for each region which is then scaled up for inference.

Obstacle Distance using Stereo-camera

Initially the stereo image is used to calculate the depth map from the input stream. This data is used to approximate the location of the obstacle detected. This data is then used for exact localization of the object after being validated by the LIDAR data.

Obstacle Distance Validation

Once object is detected on the input image, we check the frustum region that we get on extending the bounding box outwards. This helps us validate the exact location and dimensions of the object concern. We can then use this data to create a 3D bounding box around the object if necessary. Since most of the objects concern are cone. We only need the distance of the cone from the robot as well as the radius of the cone. This can be done by processing the LIDAR data. Once the obstacles have been localized, they are plotted on the occupancy grid with an exaggerated dimension, to make sure there exists a safe distance between the robot and the obstacle concern.

Sign and waypoint

The location of the Signs are just approximated using stereo vision. When the signs are within a close proximity of the robot, the track planner plans the track according to the sign. In case of a stop sign, the planner immediately terminates the movement for 10 seconds, after which it continues in the planned path. In case of a waypoint, the planner analyses the possible paths and gives higher preference to paths in accordance with the suggested signs.

Emergency stop

Failure Detection

In case of any sensor system failure, like, malfunction of Camera, LIDAR, GPS, IMU units, we need a fail-safe in place to make sure that the robot is unharmed. So in case of such failure, the ROS uses the previously saved data on the cost map, and finds a non-collision trajectory to park the robot in a safe spot. This situation has a single point of failure, i.e. the ROS.
In case of ROS master node failure, a similar program is run on the functioning Jetson nano, that uses only the camera input, to make sure the robot is safe.

Failure Points

Mechanical Sub-system Failure

Our vehicles constitutes of medium to lights weighted components, majority of weight distribution has been occupied by battery and payloads , we managed to arrange them centrally across frame ,additional weights of all components resulted in a mass centralization. Distribution was carefully estimated, so as to prevent any unfavourable or unnecessary inertia, frames are being joined with supporting plates. That further strengthens the material to withstand bending n shear forces, carefully measured so it can be subjected to moderate tension.
Although microscopic vibrations for a long interval could b a problem, which can be easily dealt using rubber padding,these will dampen the oscillating amplitudes to a great extent .
Bolts used are standard M6 anodized, with a great permissible stress value. Higher than we need , further t slot bolts are also used to tightly pack and interlock the grooves
Summing points for failure-

1) High impacts and high uniform concentration of load can create deformities
2)bolts will fail after passing their permissible stress
3)Vibration

Electrical / Electronics Sub-system Failure

- **Algorithm Error:-** The algorithm being executed by JETSON may get stuck or may show erroneous behaviour which will result in failure of the Vehicle.
- **Communication Error:-** Communication error may occur due to improper communication between JETSON micro-controllers in the Kubernetes Network which will result in erroneous instructions to the motor.
- **Temperature Error:-** The temperature of the cell may be more than the highest safe temperature which will be detected by temperature sensor and the supply to the devices like Motors, LIDAR, GPS, and Camera will be cut off by turning of the shutdown circuit which will disconnect the power from the Accumulator.
- **BMS Error:-** Any over load on the cells beyond the rated capacity will lead the BMS to turnoff the power supply from the Accumulator and bring the vehicle into halt.
- **Loose Connection Error:-** Any loose connection in the wiring harness may result in loss of power or communication with certain equipment, as a result of which the equipment will fail to perform properly.
- **Fuse Error:-** Any faulty component may demand more current than the required rated current. This will result in melting of fuse and in turn resulting in fault of vehicle.

Software Sub-system Failure

There exists scope for localization failure in case no points on the object body are detected by the lidar, in which case the best localization is to be done by the stereo vision approximation. Since the stereo vision is inaccurate as the distance increases from the camera, the scope of error increases. This is mostly overlooked as the robot approaches the object and the location of the object is rectified, but in case of object small objects or fast-moving objects, this can be a point of failure, This can be rectified by the use of higher power LIDAR devices that cover more FoV in its surroundings.

Project Budget

Component Description	Qty		Price
NVIDIA JETSON NANO	3.00	Each	₹ 28,997.00
WD SSD 250GB BLUE	1.00	Each	₹ 3,150.00
Arduino Mega 2560 Micro controller	2.00	Each	₹ 1,678.00
Raspberry Pi 4 Model B 4Gb Ram Micro Controller Board	1.00	Each	₹ 5,350.00
Sony IMX219-200 Camera - 200° FOV - 8 MP - 3280 × 2464 - Applicable for Jetson Nano	1.00	Each	₹ 1,995.00
Metal Case for NVIDIA Jetson Nano Developer Kit	3.00	Each	₹ 3,117.00
Intel® RealSense™ Depth Camera D435	1.00	Each	₹ 12,888.00
GEAR Maxon Planetary Gear head GP 42 C Ø42 mm, 3 - 15Nm, Ceramic Version Part-No.: 203116	2.00	Each	₹ 27,155.02
MAXON MOTOR Maxon EC-i 40 Ø40 mm, brushless, 100 Watt, with Hall sensors Part-No.: 496660	2.00	Each	₹ 28,273.44
SENSOR Maxon Encoder ENC 16 EASY XT, 128 cpt Part-No.: 584776	2.00	Each	₹ 16,107.12
MAXON CONTROLLER Maxon DEC Module 50/5, digital 1-Q-EC Amplifier 50 V 5 A, speed control, OEM module Part-No.: 380200	2.00	Each	₹ 7,816.54
ACCESSORY Cable with Connector Part-No.: 339380	2.00	Each	₹ 2,010.66
NOVATEL SMART ANTENNA	1.00	Each	₹ 226,872.00
SICK 3D LIDAR	1.00	Each	₹ 216,000.00
MAXON MOTORS CUSTOM DUTY	1.00	Each	₹ 5,345.00
Intel® RealSense™ Depth Camera D435 SHIPPING	1.00	Each	₹ 2,148.48
NOVATEL SMART ANTENNA SHIPPING	1.00	Each	₹ 12,600.00
SICK 3D LIDAR SHIPPING	1.00	Each	₹ 10,800.00
MAXON MOTORS SHIPPING	1.00	Each	₹ 3,539.60
TOTAL			₹6,15,842.86

Table iv. Project Budget

References:

1. Lanenet from Towards End-to-End Lane Detection: an Instance Segmentation Approach by Davy Neven Bert De Brabandere : https://arxiv.org/pdf/1802.05591.pdf

2. SlimYOLOv3: Narrower, Faster and Better for Real-Time UAV Applications by Pengyi Zhang: https://arxiv.org/ftp/arxiv/papers/1907/1907.11093.pdf

3. **Robot Operating System (ROS): The Complete Reference (Volume 3), Studies in Computational Intelligence** Released July 2, 2018.