INTELLIGENT GROUND VEHICLE COMPETITION

2015

Indian Institute of Technology Madras, India

TEAM ABHIYAN

Faculty Advisor Statement:

I hereby certify that the development of the vehicle, ABHIYAN 1.0, described in this report has been equivalent to the work involved in a senior design course. This report has been prepared by the students under my guidance.

Nitin Chandrachoodan

Dr.Nitin Chandrachoodan
Associate Professor, Department of Electrical Engineering, IIT Madras

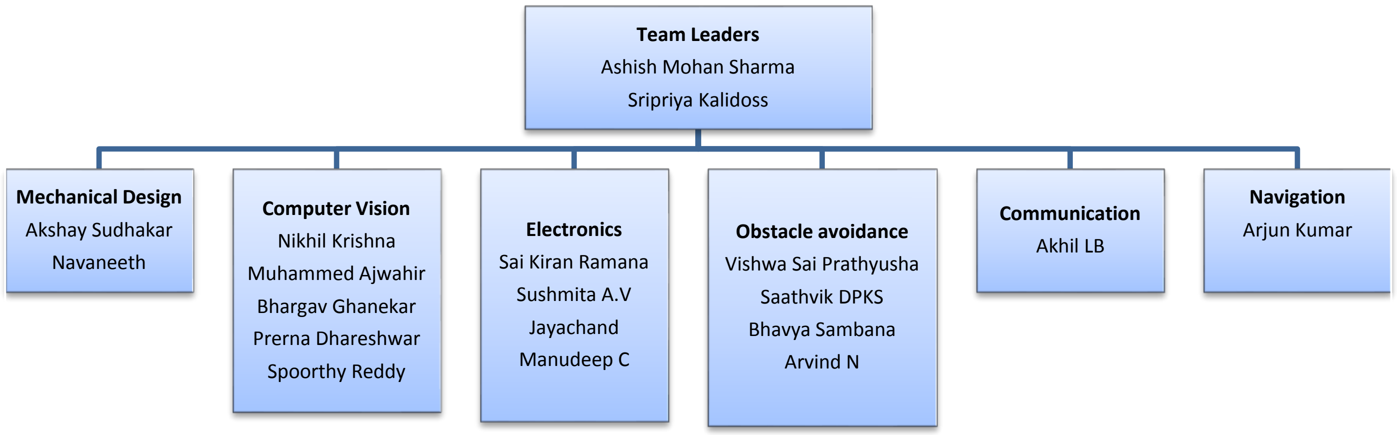
Team Members: Ashish Sharma, Arvind N, Vishwa Sai Prathyusha, Saathvik DPKS, Akhil LB, Sripriya Kalidoss, Akshay Sudhakar, Navneeth P, Susmitha A.V, Sai Ramana Kiran, Jayachand, Manudeep, Bhargav G., Muhammed Ajwahir, Nikhil Krishna, Prerna D., Spoorthy M, Bhavya Sambana, Arjun Kumar

CΦ IIT Madras Centre For Innovation

CONTENTS

1. Team Structure

The team is being incubated in the Centre for Innovation (CFI) and is being supported by funds from the Alumni of IIT Madras. The team consists of junior undergraduates belonging to various academic backgrounds viz., Computer Science, Electrical Engineering, Mechanical Engineering, Engineering Design etc. With this diversity, every member brings to the table a different set of skills and opinions, which is vital considering the multi-disciplinary nature of the project.



The tasks in hand are divided between 5 modules:

- Mechanical Module
- Electronics
- Navigation
- Communication
- Obstacle avoidance
- Computer vision

2. Mechanical Module

Introduction

The weight and dimensions of the payload were the primal considerations during the course of the mechanical design. Being the heaviest part of the vehicle it was ensured low placement and CG as near to center of vehicle as possible. The shape and structure of the frame were decided after rigorous discussion and testing considering the distribution of different components to be placed on it. The bot will be in contact with 3 points on the ground at all times. So, given any terrain, the bot will always find a plane of motion. Air-filled tyres are used for locomotion such that their pressure can be varied to dampen unwanted vibrations.

Design Methodology

1. Drive train
   The drive train is very simple and effectively designed. The motors are placed near the center of the vehicle, while the drive shaft and motors are connected using flange coupling arrangement. This arrangement reflects the design for easy assembly and disassembly in the vehicle. The coupling arrangement is attached to the frame using ball bearings on either side of the wheel, which provide support for smoothness and help avoid friction and bending moments.

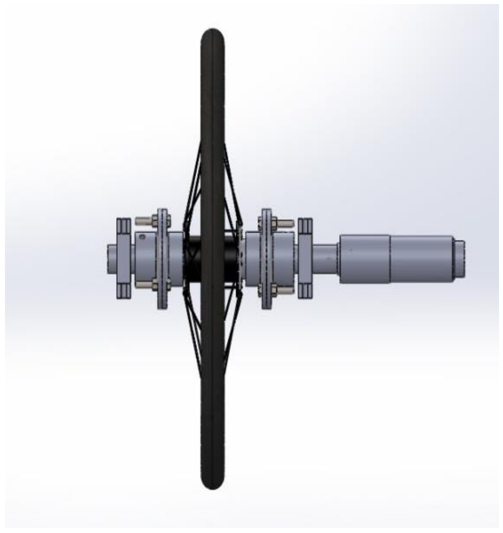


Fig.1 Drivetrain arrangement

2. Frame
   The whole chassis of the vehicle is made using Aluminium extrusion channels. The main problems with the previous prototype (Abhiyan 1.0) were the use of Aluminium L-channels and the resulting instability of the structure, which led to errors in the readings of various sensors, which led to complications in the navigation and obstacle avoidance modules. Aluminium extrusion channels are commercially designed for modularity and ease of assembly and disassembly of the complete frame. Thus, the fabrication time and efficiency of the vehicle were reduced considerably. The extrusion channels can be joined using angle brackets at right-angle joints. We can also tap threads into the channels and use T-nuts and Button-head bolts to fasten 2 different channels.

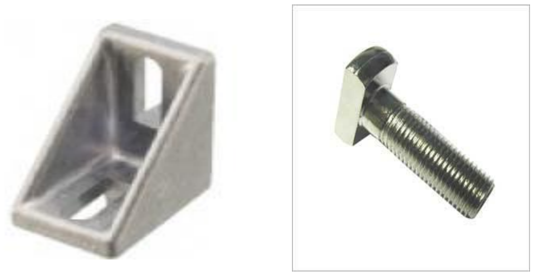


Fig.2 a) Angle bracket b) T-bolt

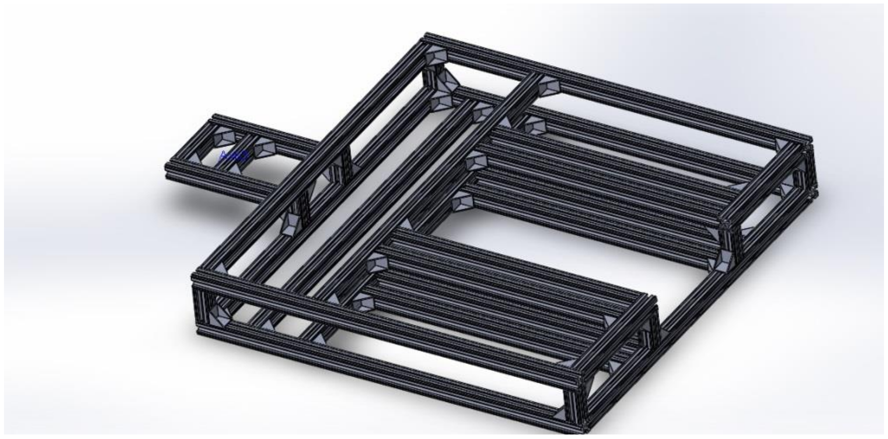


Fig.3 The complete frame designed using aluminium extrusion channels. Notice the right-angle joints are connected using angle brackets or by tapping the channels and using T-bolts.

3. Steering
   Being a completely autonomous vehicle, no manual input is allowed once the vehicle is on track. The differential drive was chosen as the steering methodology because of its simplicity and efficacy for our requirements. The vehicle has two powered wheels and one castor. The wheels are powered by one motor each. The RPM of the motors is controlled independently by the microcontroller. By varying the RPM of the motors the steering angle and the direction of heading of the vehicle can be changed. Internal encoders fitted with the motors are being used for precise motor control and odometry calculations for navigation purposes.

4. Castor positioning
   The position of the castor plays a vital role in the dynamic orientation of the vehicle. Keeping the castor in the front would make the robot sway in the direction of the castor, which is theoretically controlled by the motion of the powered wheels. But after extensive testing it was observed that on rough surfaces the castor generates its own direction which makes the vehicle deviate from the desired path. Hence it was decided to use 2 front powered wheels with a rear wheel castor.

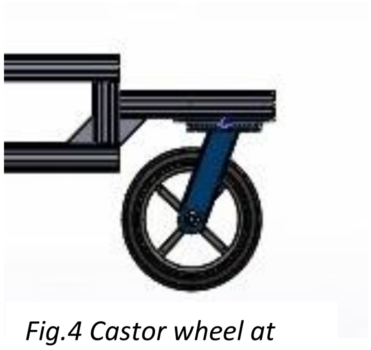


Fig.4 Castor wheel at rear of the vehicle

Design Process

Computer Aided Design: The 3-D modelling software SOLIDWORKS by Dassault Systems was used for CAD modelling purposes. The advantage of using this software is that it reduces physical design iterations which can save time in the design process. There are also provisions for stress analysis on the design to avoid material failures.

Payload: The vehicle is required to run with a payload of 10 kg weight. The design and component distribution has largely been influenced by the presence of the payload. A low and centralized center of-gravity has been maintained to provide maximum dynamic stability to the vehicle.
Vehicle Configuration: After deep deliberation and analysis, it has been determined that two wheeled front powered vehicle with a single rear castor is required for locomotion, providing greater stability while turning. This weight distribution of the components and low speeds ensure ground contact and enough normal forces on suspensions at all times.
Fabrication: The entire chassis was assembled using standard Aluminium extrusion channels from Alstrut Inc. This improved the structural integrity of the vehicle and reduced mechanical errors in readings from the various sensors.

Torque requirements

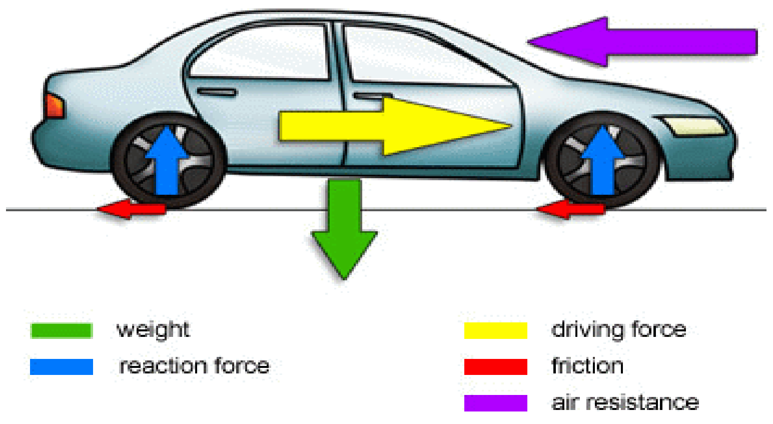


Fig.5 Free body diagram indicating the forces acting on the vehicle

The major forces acting on the vehicle are-

- Aerodynamic force: These are the drag forces that act on vehicle due to wind resistance. The forces are directly proportional to the square of velocity of the vehicle.

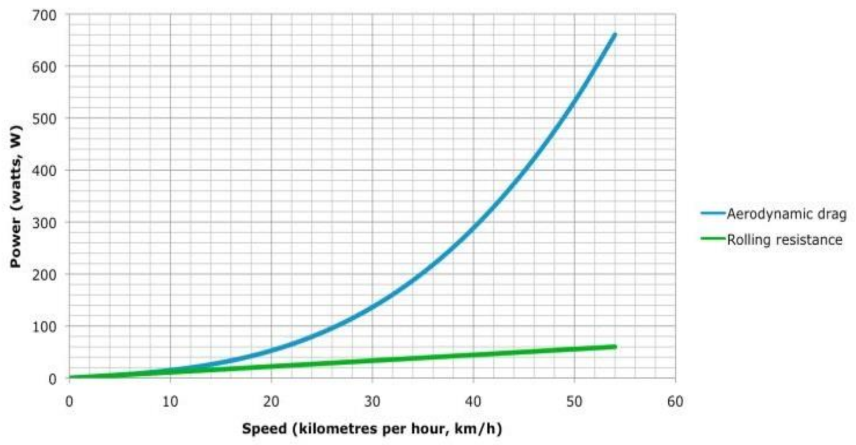


Fig.6 Plot of aerodynamic drag power vs. speed

From the plot and the equation it can be deduced that the air drag for our requirement is very low and can be neglected.

- Rolling resistance

The rolling resistance is the friction that the vehicle experiences due to hysteresis in the runner tires. The Rolling resistance is a constant value and is computed by

$$R_r = \mu_r \times N$$
$$\tau = r \times R_r$$

- R_r = Rolling Resistance on Rear wheels
- μ_r = Coefficient of rollong friction between rubber and ground
- N = Normal
- r = *Radius of the driving wheel*

Variable	Value
μ_r	0.35
N	180 N
R_r	24.5 N
r	203 mm

Motor specifications

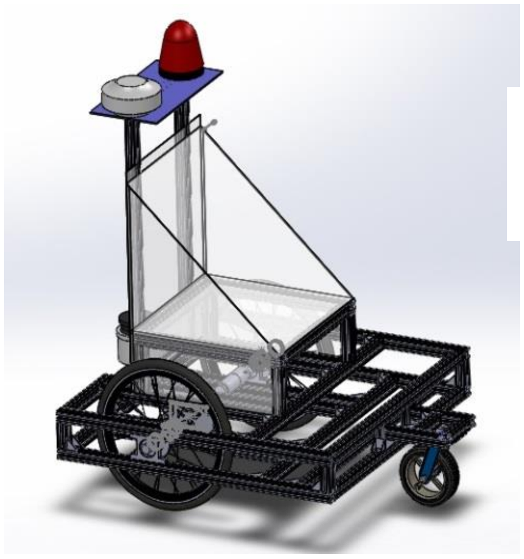
We need to achieve a minimum speed of 1mile per hour and a maximum speed of 5 mile per hour. Motor specifications were calculated using a max speed of 3.9kmph and an average speed of 2.4 kmph. Maxon Brushless DC motors are used with the ESCON Module 50/5 Servo Controller, which is a small-sized, powerful 4-quadrant PWM control for these type of magnet-activated motors.

No load speed	7580rpm
No load current	137mA
Stall torque	2280 mNm
Max. Efficiency	91%
Operating Voltage	24V

Fig. 7 MAXON DC Motor

Fabrication and Assembly

Aluminium extrusion channels are used to build the chassis. The channels are connected using angle brackets or T-joints. The motors were secured to chassis using vertical clamps to reduce the vibrations and proper transmission of power to the wheels. Parts like bearing-casing, couplers, intermediate shafts, rear wheels assembly were manufactured using lathe and milling machines. Specific mounts were designed and fabricated for the multiple sensors like LiDAR, GPS, Camera and a separate shelf for placing of the laptop computer. Enclosure of the entire vehicle for rain proofing purposes will be done using acryclic sheets.



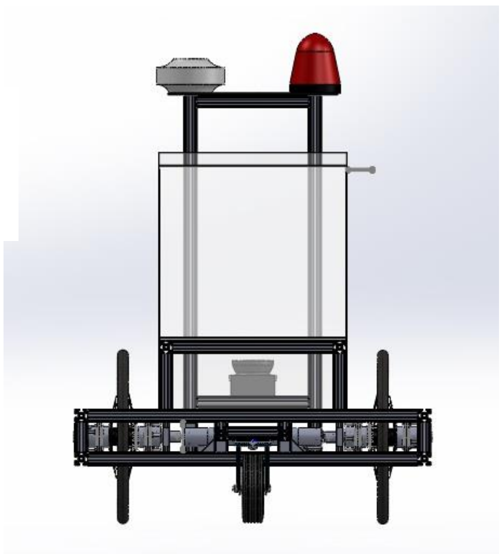


Fig.8 Final vehicle
a) Isometric view
b) Front view

3. Electronics module

Power System

Among all the sensors in the bot, the LiDAR, GPS sensor and the motors consume the maximum power. Lithium polymer batteries are used as sources of power in the vehicle.
The sources of power in the circuit are divided into two:

a. Main Source: Five 3cell 30C 8000mAh batteries explicitly to power the motors, GPS and LiDAR and safety light.
b. Auxiliary Source: One 3cell 30C 8000mAh battery to power the entire circuit.

Wireless control

The Auxiliary power source is connected through the RF module on the bot. This RF Module helps to wirelessly control the circuit by switching it ON/OFF using a remote control. This serves as the Wireless E-stop, added to the Mechanical E-stop (Push-button) on the vehicle.

Odometry: Control and PID Tuning

MAXON motors have internal encoders with 5376 Ticks per rotation which is very high for a normal microcontroller unit to count the ticks appropriately. Hence, we use the Arduino Due QEI to get Odometry data from wheel encoders which will be then used by the path planner in ROS. ESCON motor drivers are used along with MAXON motors to control the speed of the motor and regulate the current to the motors depending on the load. They contain internal circuitry that can power and control the operation of the motors, according to the PWM signal sent by the Arduino. To configure these motor drivers, the ESCON Studio software is used to tune and configure the motors accordingly in order to achieve the desired set speed with ease. The tuning is done at high frequency through inbuilt PID controller which follows basic Ziegler-Nichols Algorithm.

Microcontroller Unit
The Arduino Due was preferred as the MCU because of its QEI (Quadrature Encoder Interface) which is designed to count motor encoder ticks. Two Arduino Due are used since any one of them individually cannot handle the encoder counts in the motors. I2C serial communication interface is implemented between the 2 Arduinos to transfer data between them.
1)Master MCU: This powers and transmits data from the compass, controls the bot speed with appropriate PWM signals and retrieves odometry data from the Slave MCU.
2) Slave MCU: Its sole purpose is to get the odometry data from the wheel encoders and send it to Master Arduino by serial communication.

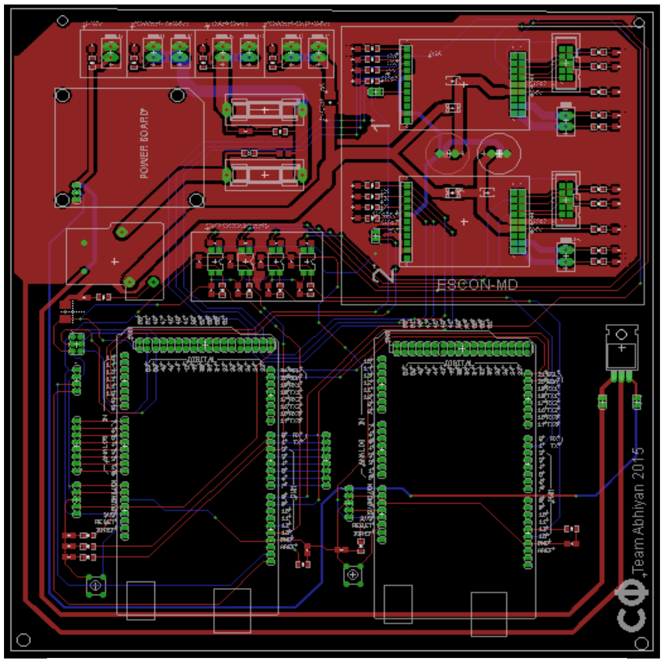


Fig. 9 PCB designed using Eagle CAD

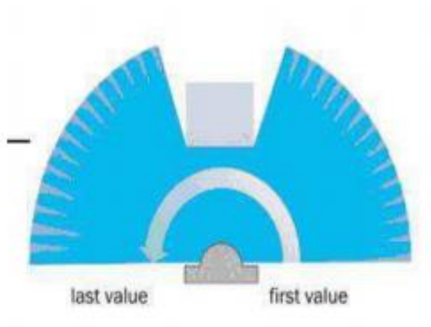
Since the digital I/O pins in the Arduino can only take till 3.3V, there is a risk of burning the MCU while interfacing with sensors like the wheel encoders. To prevent this, optocouplers are used to step down the voltage signal of the encoder before providing it to the Due.

Eagle CAD is used to design the PCB. Basically, the circuit board is virtually partitioned into two parts -
- Main Board
- Power board.

The main board consists of the Arduino Due shields, ESCON motor driver shields and optocoupler shields. Fuses are used to protect various components of the circuit from shorting and current surge.

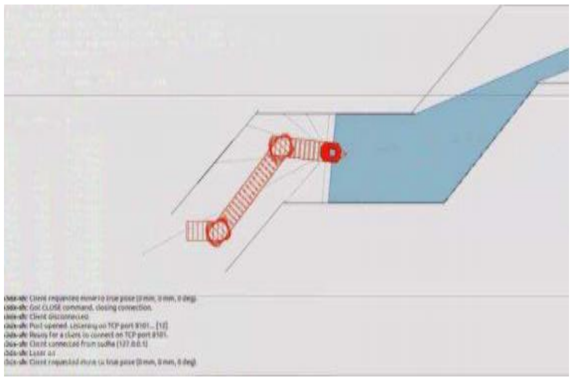
4. Obstacle Avoidance

Fig.10 Sample LiDAR scan

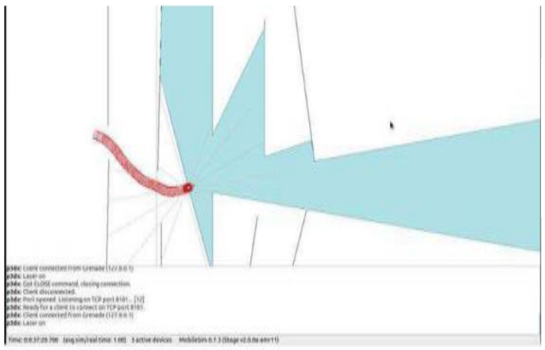


One of the main functionalities of this ground vehicle is to perceive the environment, and negotiate outdoor obstacles. Obstacle avoidance is, hence, the most fundamental and essential part of the vehicle's navigation system.
The crucial component for this is the sensor, which is chosen to be a LiDAR (Light Detection and Ranging Device). LiDAR is a laser based ranging device that scans the surroundings in 2D plane in front of it to detect objects and evaluate their positions. It, basically, gives a polar profile through which the distance of the nearest obstacle at any angular position can be measured.

Earlier, the basic algorithm was simulated and tested in various software (MobileSim of ARIA), the results for which have been summarized in the following section. The figure above shows the data obtained in a LaserScan. The whole region is segmented into different regions called as valleys and peaks. Valleys represent no obstacle region and peaks represent the region in which obstacle is present. Hence, at any instant the algorithm looks for all the valleys and chooses a valley closest to the goal position with a constraint that the valley width is greater than the robots width. MobileSim is a software on ARIA (Advanced Robotics Interface for Applications) platform of mobile robots that is being used for simulating the basic algorithm.



```
[illegible]: Client requested move to true pose (0 mm, 0 mm, 0 deg)
[illegible]: Got CLOSE command, closing connection.
[illegible]: Client disconnected.
[illegible]: Port opened. Listening on TCP port 8101... [12]
[illegible]: Ready for a client to connect on TCP port 8101.
[illegible]: Client connected from sudha (127.0.0.1)
[illegible]: Laser on
[illegible]: Client requested move to true pose (0 mm, 0 mm, 0 deg)
```



```
p3dx: Client connected from Grenade (127.0.0.1)
p3dx: Laser on
p3dx: Got CLOSE command, closing connection.
p3dx: Client disconnected.
p3dx: Port opened. Listening on TCP port 8101... [12]
p3dx: Ready for a client to connect on TCP port 8101.
p3dx: Client connected from Grenade (127.0.0.1)
p3dx: Laser on
Time: 0:0:37:28.700   (avg sim/real time: 1.00)   3 active devices   MobileSim 0.7.3 (Stage v2.0.0a-amr11)
```

Fig. 11 Simulation on MobileSim Software

<u>Limitations</u> -

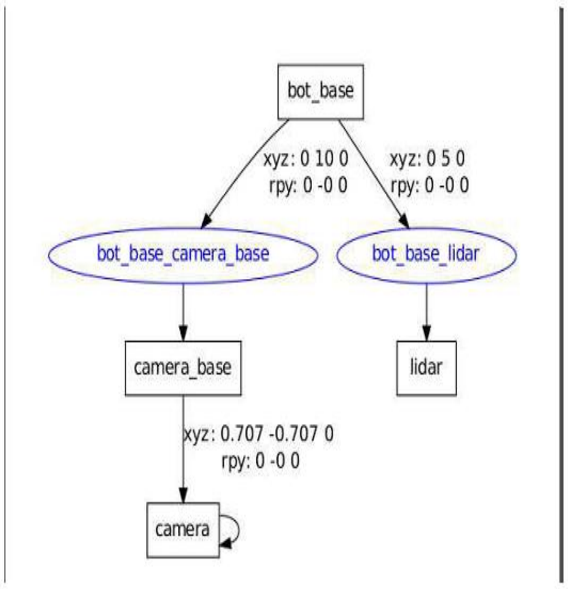
- The present mapping software cannot be used for complicated environments
- The motion control and other functions are all in-built, which cannot be customized as per our requirements
- Direct integration with other modules and ROS is not possible.

Current Control and Operating System

Moving to an advanced stage, it has been decided to use ROS (Robotic Operating System) and the in-built algorithms available with its packages for the navigation of the robot. The current control and operating system in Abhiyan robot is based majorly on ROS.

ROS is an operating system with a set of libraries and software for various robotic applications. It also covers various algorithms for many programs, provides visualization software for simulations, device drivers for various sensors, and also unique opportunity of synchronization with various applications at a

Fig. 12 Robot configuration



IIT Madras Centre For Innovation

time. That allows for better and easier ways of integration of different modules.

In such huge set of packages available, the package called ‘Navigation Stack’ is being used for this robot. Navigation stack takes in input data from all the sensors attached to the robot, processes them in different appropriate nodes simultaneously or sequentially, and gives the values of the velocities to the robot to follow as an output.

<u>Working of Navigation Stack</u>

Input:

Robot Configuration - This includes transform configuration and robot setup, wherein the physical shape, size and dimensions of the robot are set up. With this information about the robot, all the frames available on are identified. This transform considers all the coordinate frames and uses this transform to interpret data from various sensors appropriately.

Sensor Information - Data from different sensors is obtained and taken in form of messages. There are a set of kinds of messages in which the data can be published and subscribed by different nodes whenever required.

A view of the frames of the robot, where LiDAR (base_laser), Camera (laser_frame) are attached to the main robot base (base_link) receiving odometry readings (odom), also connected to IMU (imu_frame). Also shows the parameters in which the sensor data is being received.

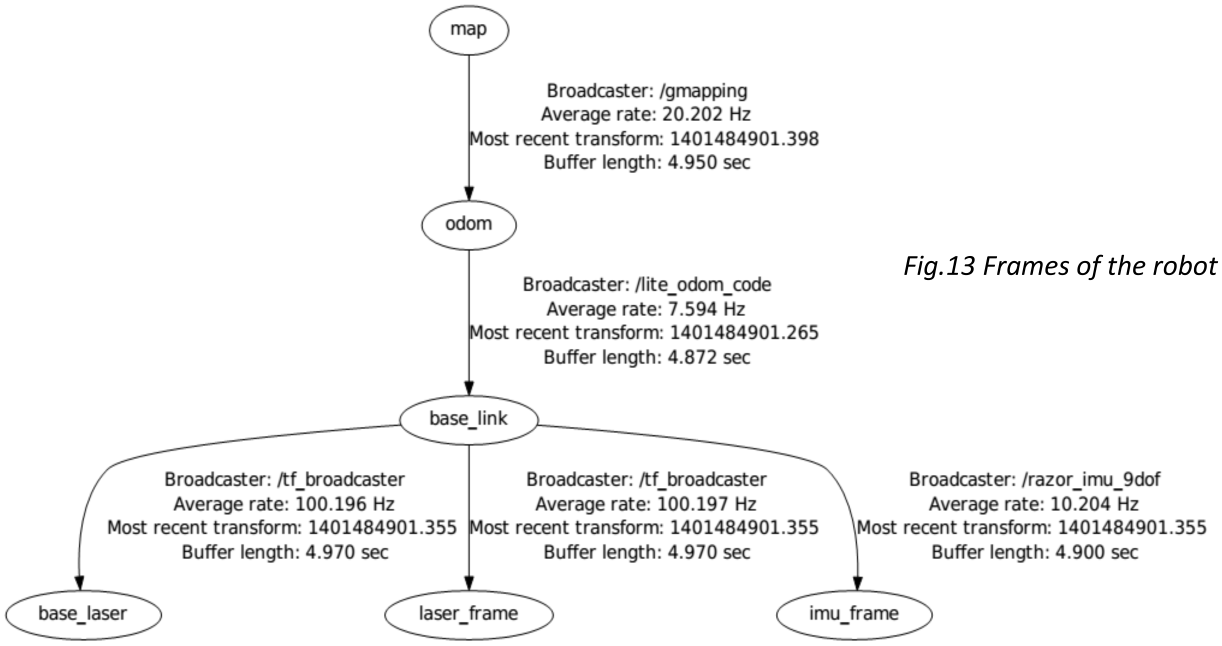


Fig.13 Frames of the robot

Process and Outputs:

The navigation stack with all the parameters in the .yaml files for robot configuration, with all the inputs from different sensors published/subscribed by different nodes, maps are generated with specified goals to reach. Then, by using SLAM(Simultaneous Localization and Mapping) in gmapping package on a dynamic map or by using AMCL(Adaptive Monte Carlo Localization), the pose of the robot is traced and localized. After the localization of the robot, according to its position with respect to the target location, the robot is given velocity commands (/cmd_vel) to reach the goal.

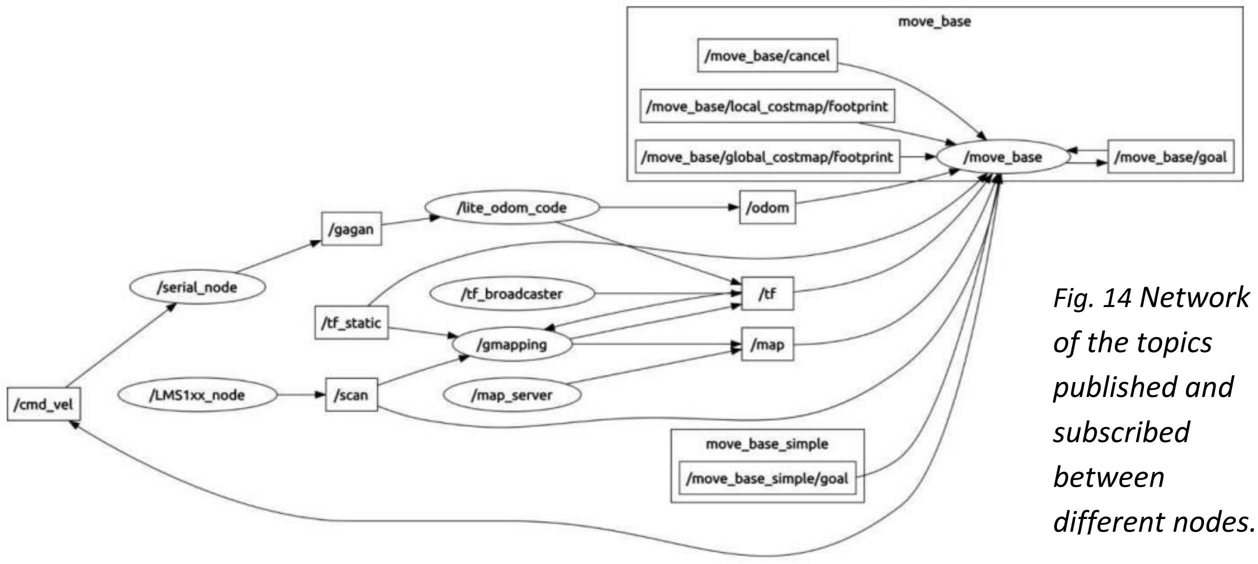


Fig. 14 Network of the topics published and subscribed between different nodes.

5. Image processing

Introduction

Major challenges in the problem statement include:

- Detection of lanes and simulate potholes in the arena
- Detection of red and blue flags and navigate the vehicle accordingly

The goals of this module were to effectively isolate the lanes and potholes and feed this data in some format to the path planner for further processing. Also, it was required to obtain the positions of the red and blue flags in the arena and transfer the data to ROS for planning and navigation.

For lane-and-pothole detection (Camera 1), a Logitech C310 camera (640X480) is used to capture the view of the arena. The camera is mounted at a height of 43.3cm from the ground. For flag detection (Camera 2), a camera of the same model is used but at a height of 100cm from the ground.

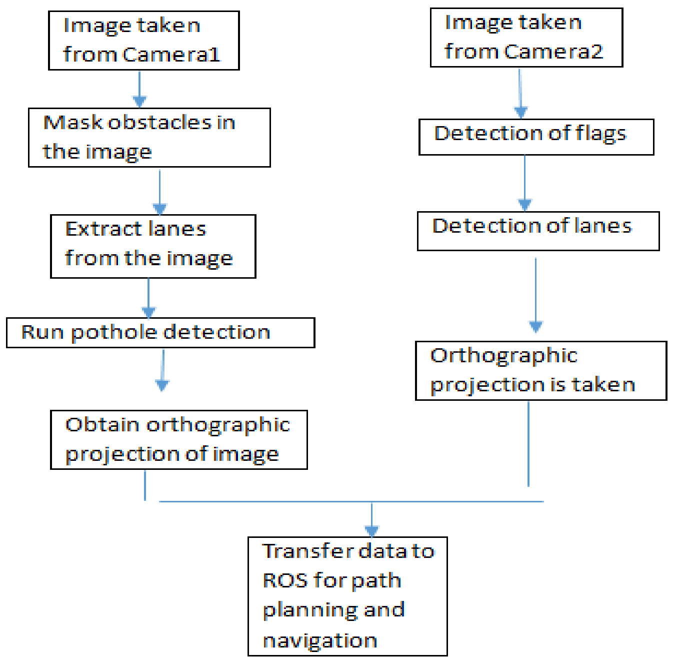


Image taken from Camera1
Mask obstacles in the image
Extract lanes from the image
Run pothole detection
Obtain orthographic projection of image
Image taken from Camera2
Detection of flags
Detection of lanes
Orthographic projection is taken
Transfer data to ROS for path planning and navigation

Fig. 15 Breakdown of the algorithms used. Scripts implemented in C++ using OpenCV libraries.

Bird's eye view transformation

The orthographic view of the camera is computed using the formula:

$$A = HB$$

Where $A \equiv$ orthographic/final computed image;
$B \equiv$ initial image captured by the camera;
$H \equiv$ Homography matrix;

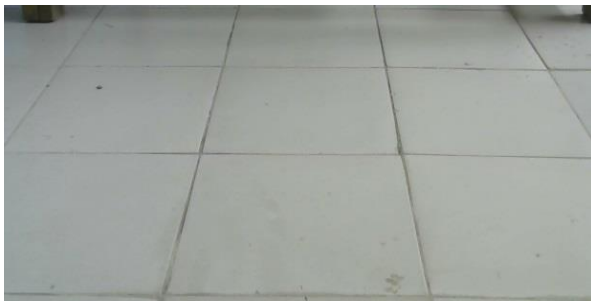


Fig.16 a) Normal camera view

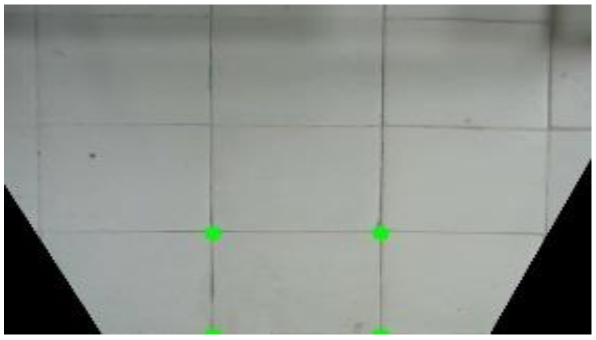


Fig. 16 b) Orthographic view

The Homography Matrix is calculated by training the camera for various heights and angles, so as to minimize the errors obtained. After rigorous testing, it was finally settled to have the camera mounted at a height of 43.3cm above the ground, tilted to an angle of -20^{o} with the horizontal. This is also reduces the blind spot distance, so as to capture the parts of the lane which are close by. The blind spot distance is approximately 67cm from the front-end of the vehicle.

Obstacle masking

The lane detection algorithm requires the obstacles to be masked/ignored from the image for efficient computation. Hence, this is done using HSV segmentation of the blue channel of the input image and applying an automated thresholding algorithm which takes the minimum peak in the histogram as the thresholding characteristic.

- Image is converted from RGB to HSV
- Orange parts of the image are thresholded to white in a binary image.
- Convolution with a vertical kernel is done to mask out striped barrels properly.
- Morphological opening and closing algorithms used to smoothen the contours
- Subtract the final image from original image to mask out the obstacles
- A contour-area based filter can also be used to get rid of the blobs after convex hulling.

Lane detection

The algorithm for extracting lanes from the grassy arena includes converting the color space to HSV for better color segmentation. The obtained image is processed by the Canny Edge Detection Algorithm before running it through Probabilistic Hough Transform. The voting procedure is set up to maximize the number of lines of a minimum length that can be generated in the image.

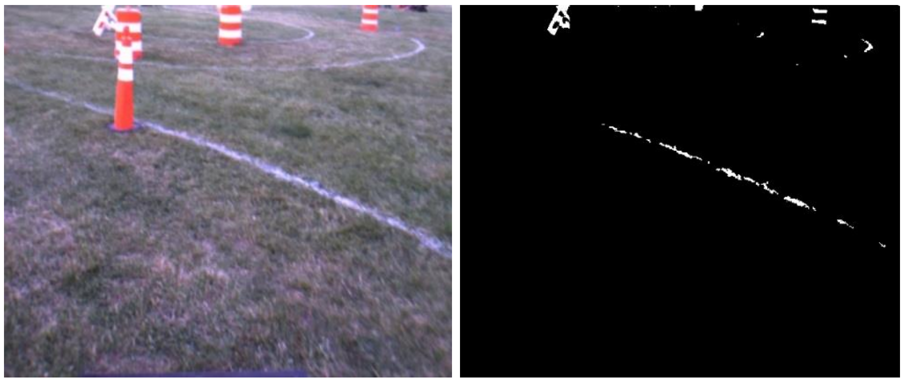


Fig. 17 Lane detection after masking obstacles
a) Original image
b) Final image after processing

Pothole detection

An essential part of the problem statement is to detect simulated potholes (white in color). The circular potholes are actually seen as ellipses in the non-orthographic, i.e. the camera view.
We take the image processed through the obstacle masking and lane detection algorithms. A median filter is then applied, followed by canny edge detection. The obtained contours are fitted to ellipses using perimeter thresholding. These ellipses are masked as potholes.

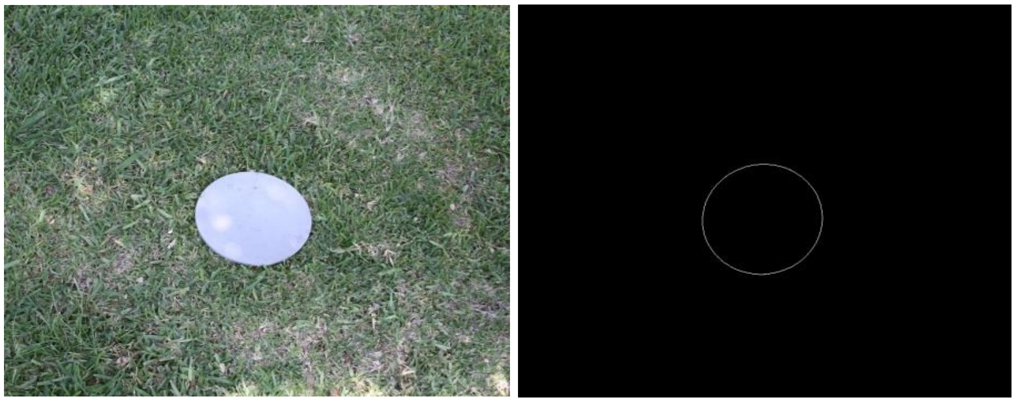


Fig. 18 Simulated pothole detection a) Simulated pothole b) After processing

Flag detection

Flag detection is done using Camera2 placed above Camera 1.
The centroids of the triangular red and blue flags are roughly in a plane parallel to the ground. Hence, a homography matrix calibrated for a height of 100cm above the ground is used so as to get the correct orthographic view.

Red (Grainger3LUK2) and Blue (Grainger3LUK4) flags are detected using HSV segmentation. These flags are fitted into a polygon, whose centers are located and joined by best-fitting lines. The objective is for the vehicle to stay to the left of the red flags and to the right of the blue flags. Thus, we can create a wall in the infeasible regions of travel for the bot, so that the feasible regions remain open. This final image depicts the regions of open allowable travel for the bot.

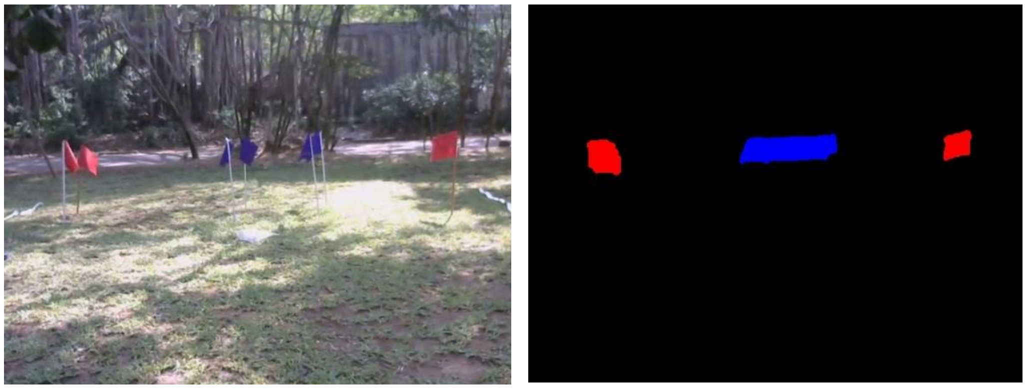


Fig. 19 Flag detection a) Original image b) Red and blue flags detected as continuous contours

ROS Integration

The lanes, obtained above, are converted into the form of LaserScan data. This conversion is needed because the ROS Gmapping package can accept data inputs only in the form of LaserScan data or Point Cloud data to generate maps. After converting to LaserScan data, the CV images are converted to ROS image messages using cv_bridge package, so that the images can be published, and accessed by the ROS Gmapping package, which visualizes the lanes as walls/obstacles to be avoided. This data is fed into the navigation stack, along with the LiDAR, GPS and odometry data for path planning and autonomous navigation. The walls obtained in the flag detection script can also be passed as a LaserScan to ROS, which will run it through similar procedures to depict the possible paths of navigation for the bot. The simulated potholes will also be seen as walls, applying the same methods to these as well.

Further challenges in these algorithms include noise removal in video feed, processing capabilities of the computer and ambient conditions of the track.

6. Communication

The JAUS challenge requires us to implement JAUS services on our machine and make it interoperable. We have used the JAUS++ SDK for this purpose. JAUS++ is an Open Source implementation of JAUS in C++. It implements the current versions of the SAE standard. JAUS++ includes complete message libraries for both the Core and Mobility service sets, and interfaces for the most common services. JAUS++ is Object-oriented in design and provides threads in the background to send and receive messages. Every JAUS component has a subsystem ID and a node ID. The assignment of these IDs is static. For transport, JAUS over UDP transport protocol will be used as specified in AS5669A.
The UGV first tries to discover the calling device and tries to establish a connection with that node. It keeps listening to that node for queries. As soon as a query is received, it processes the type of query it is and decides which action to take. If a query for velocity or GPS is made, the last value of the required component published from ROS is read and updated in the JAUS system navigation points and fed back to ROS. JAUS++ provides us an interface to update different components and since it runs threads to receive and send messages in the standards mentioned by SAE, we don't have to worry about them while coding it. Since a reliable transmission is needed, JAUS is implemented on TCP.
The following diagram shows the class inheritance diagram for services within JAUS++. It shows the Core Services and a few Mobility Services available.

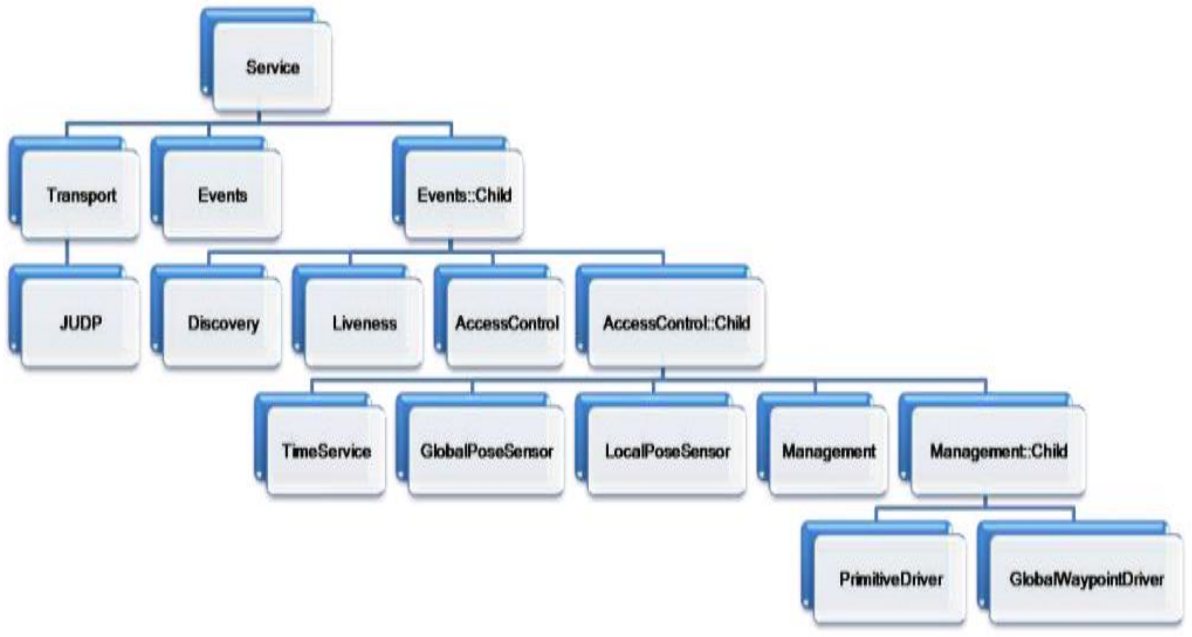


Fig. 20 JAUS system architecture

7. Navigation system

Introduction

A vital system for any ground vehicle to make an accurate decision about its heading and navigate from initial position to final position. Here, we have used global coordinate system i.e. Global positioning system (GPS) for localization in the vehicle's frame of reference. Apart from this, Inertial Measurement Unit (IMU) is also used to obtain localization in vehicle's frame. This provides another input for vehicle's movement to know its position and orientation accurately. To know the vehicle's heading with respect to the Earth's magnetic field, combination of triple axis magnetometer and triple axis accelerometer is used. Combination of these two sensors provides tilt compensated values of vehicle's heading.

GPS Hemisphere A101 Smart antenna:

This Antenna uses GNSS to locate its position at an update rate of 20 Hz. It can also be used as DGPS using SBAS and NMEA 0183 protocol. Accuracy of the device is around 50cm.
Software like Pocketmax 3 or SLXMon is used to calibrate the GPS device using the Microsoft Windows platform. To get faster satellite locks, the masking angle can be reduced. DGPS is used to get faster and more accurate values.
We are using the GPSD client to publish the Co-Ordinate data and integrate it into ROS.
GPS readings are obtained and converted to link vehicle's local coordinate system to the Global coordinate system using Spherical Polar coordinates. The heading angle of the vehicle, in

local coordinate system, with respect to Earth magnetic field direction (N-S) is obtained from the compass. We have used GPS system and compass to determine the angular rotation of the vehicle .This angle of rotation can be fed to the wheel encoder using algorithm to rotate precisely with same amount as calculated. Here Inertial Measurement Unit (IMU) helps in tracking precise angular rotation of vehicle and its motion.

Implementation in ROS:

With respect to ROS, the navigation module includes some basic tasks to be done:

1) Giving the bot a direction toward next GPS waypoint
2) Sending goal points to navigation stack
3) Providing the remaining modules with necessary data (this includes providing odometry data to robot_pose_ekf package for estimating bot's current position).

Obtain GPS values: The gpsd daemon running on the Ubuntu looks into device file onto which the values are written by the GPS. Using telnet we publish these values onto a port. The gpsd_client node running in ros listens to this port and takes in the value and publishes it by a topic named /fix.

Give direction to bot: The gps_com_filter node implemented in the gps_t package takes in an input from both the compass (for calculating bot heading) and the GPS data (for calculating direction towards GPS waypoint) and depending on them calculates a direction which is published for the remaining modules to make a decision based upon.

Goal points from GPS: Navigation stack requires goal points specified on base frame for making a movement. The gpsgoal node present in the package simple_navigation_goals sends goal points in bots base frame to the navigation stack. These goal points are calculated based on the present GPS coordinates and the bots heading.

Odometry data: The robot_pose_ekf package requires odometry data for cal culating present pose of the bot. This is calculated by the gps_common node present in gps_umd package
The odometry data calculated is published by the topic name.
Block diagram for above mentioned steps:

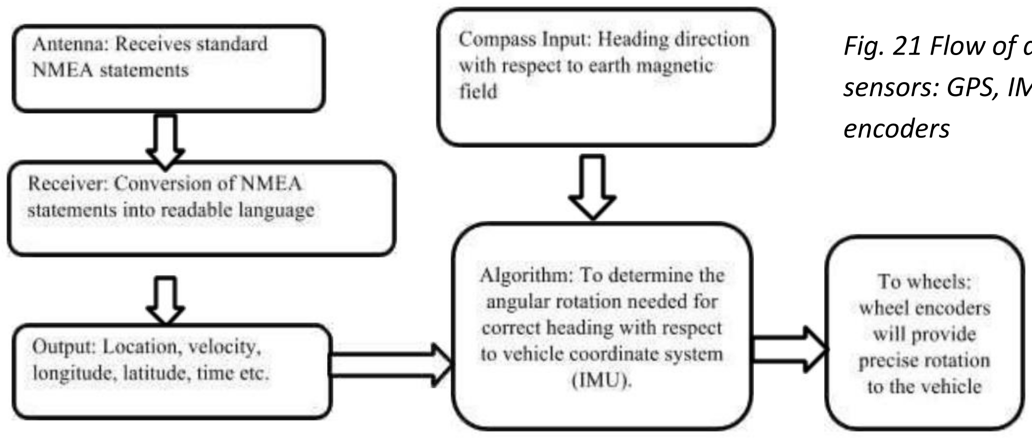

Antenna: Receives standard NMEA statements
Receiver: Conversion of NMEA statements into readable language
Output: Location, velocity, longitude, latitude, time etc.
Compass Input: Heading direction with respect to earth magnetic field
Algorithm: To determine the angular rotation needed for correct heading with respect to vehicle coordinate system (IMU).
To wheels: wheel encoders will provide precise rotation to the vehicle

Fig. 21 Flow of data from odometry sensors: GPS, IMU, Compass and Wheel encoders

8. System Integration

All sensor data and control systems are integrated using the Robot Operating System platform. This graph represents the publisher-subscriber data flow model for the Abhiyan robot.

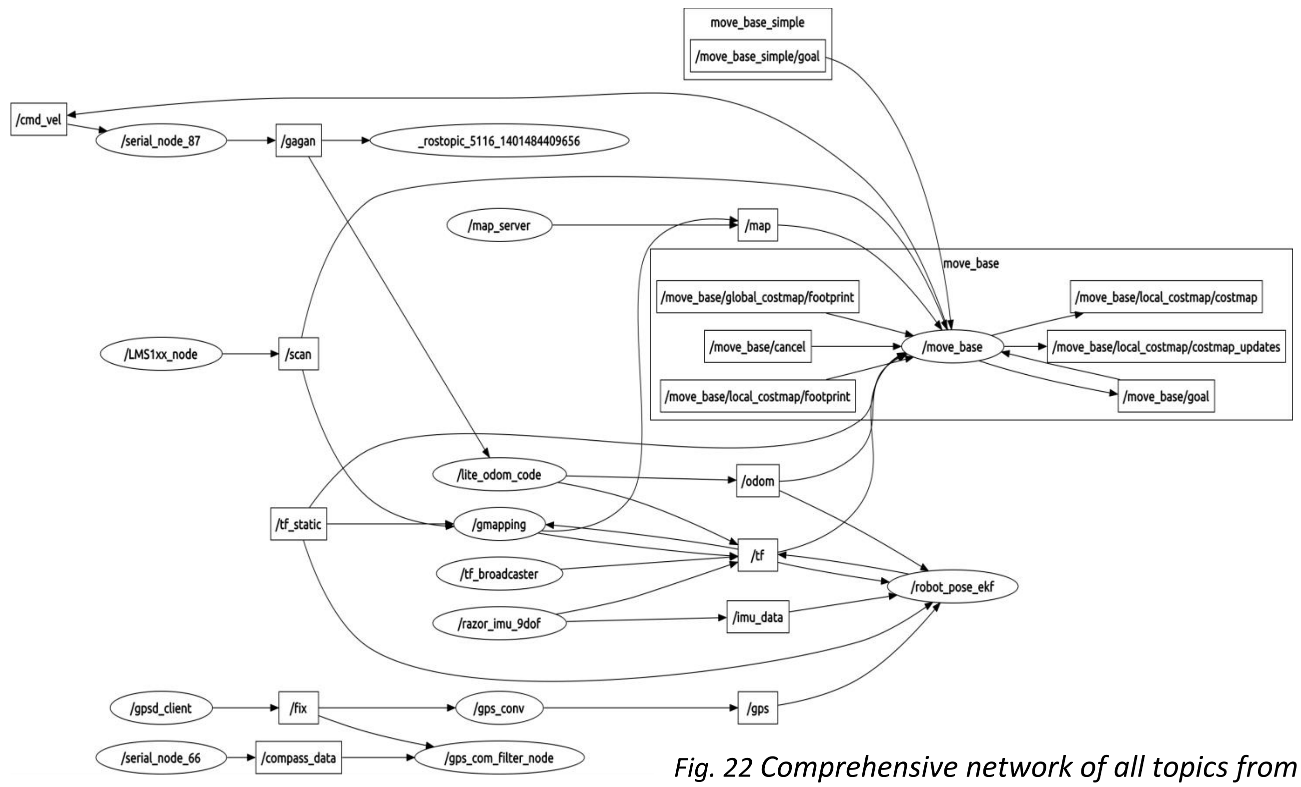


Fig. 22 Comprehensive network of all topics from the final integrated multi-sensor system of all the modules together.

9. What makes the project unique?

Our Intelligent Ground Vehicle has the ability to traverse rugged and dangerous terrains that any human will find difficult. Also, the IGV can move from one goal point to another with considerable accuracy and avoid obstacles within its range of motion. Features of lane detection and flag detection are implemented using a camera positioned on the bot. These features can be easily adapted and its uses are multi-varied and distinct. The prototype when implemented on large scale can be used for sting operations by the military. The Image processing technology used for path following can be extended for many Human Computer interface technologies which can be used in the consumer market. On the whole the complete integration of hardware and software makes it very versatile for different applications. The technology we are developing is an intelligent ground vehicle. It is a vehicle that can be used in semi–rugged terrains for military operations, mitigation, surveillance and sample collection. Capable of traversing over sand, gravel and grass, the vehicle realizes autonomous navigation and detects obstacles through laser ranging. Algorithms developed to automatically generate waypoints, will guide the vehicle in unidentified regions. Manual control of the vehicle is achieved using wireless communication. The vehicle can thus be controlled from a remote base station. Together with an image processing module in place, the vehicle is capable of independently navigating in a new environment. It is also capable of delivering a live video, recorded by the on-board camera at a remote location. The technologies involved in the project are those of emerging industries today, with great opportunities for breakthroughs and innovation.

10. Cost Analysis

Part/Electronics	**Company & Model No.**	**Description**	**Amount (In USD)**
Laser Range Finder	SICK AG & LMS 111	2D LaserScan sensor	3417.5
Inertial Measurement Sensor	Sparkfun & Razor	Three Axis Gyro, Accelerometer and Magnetometer	209
Global Positioning System	Hemisphere GNSS & A101	GPS receiver module	2295
Digital Compass	Sparkfun & Adafruit HMMC5883L	Tilt compensated compass	29.10
Camera(2 No.s)	Logitech & C310	HD720P Webcam	54.8
DC Geared motors	Maxon	DC geared motor with internal wheel encoders	N/A
Batteries	ZIPPY Flightmaxx	Lithium Polymer batteries	234
Fabrication	N/A	N/A	374
Circuitry	N/A	Electronics & PCB printing	796
Total Cost			7409.4 USD

11.Conclusion and Acknowledgements

We would like to thank Centre for Innovation, IIT Madras for being our chief patron and providing us major funding for vehicle construction, travel and logistics. We would also like to thank our Faculty Advisor, Dr. Nitin Chandrachoodan, for his unshaken faith in our team, and the Co-Curricular Advisor, Dr. Mahesh Panchagnula for his support and guidance. Lastly, we are grateful to the Centre for Industrial Consultancy and Sponsored Research, IIT Madras for helping us with procuring the imported components with relative ease.

We hope that this report is a clear testimony of the work we have been doing and the challenges we have resolved in the building of an intelligent ground vehicle.