Published by :
http://www.ijert.org

International Journal of Engineering Research & Technology (IJERT)
ISSN: 2278-0181
Vol. 8 Issue 12, December-2019

Use of Global Mapping Architecture for Implementation of All Terrain Robot Movement

Rishabh Kumar [1], Shivam Sharma [1], Ashish Singh [1], Mohit Bihany [1], Manas Ranjan Swain [1]
Harimohan Jha [2], Debashish Chakravarty [2].
[1]Birla Institute of Technology, Mesra.
[2]Indian Institute of Technology, Kharagpur.

***Abstract*— The aim of this paper is to conceptualize, design and implement the working of a self-operated vehicle that is capable of fulfilling transportation and delivery needs of its users for ease of operations and increased safety. The system operates by receiving the desired destination location details from the user. The user just needs to click on the map provided by OSM to give the destination location to the robotic vehicle remotely over the network. The system uses Robot Operating System (ROS) as a middleware sitting on top of the integrated sensor circuit placed on all terrain mechanical chassis. This vehicle provides single point access of control for the system of the vehicle from anywhere on the same local network. The OSM map application is in the master server, which ensures that not everyone can have the access to the vehicle's control unit. This work has been demonstrated on a small robotic vehicle designed for all terrain movement**

Keywords*— *All terrain vehicle, ROS, Open Street Maps, GPS, Control system.

1.0 INTRODUCTION

Self-driven or autonomous vehicles have become a cutting-edge research topic in robotics and automation domain [1-3]. They are capable of fulfilling the traditional vehicles' capabilities without human intervention. Autonomous vehicles perceive their environment with sensors deployed on them. These sensors mainly provide information which are processed further to help in safe navigation of vehicle to reach desired destination.

The prototype vehicle mentioned in this paper is autonomously driven according to destination mentioned by the user. Concurrently, vehicle needs to recognize its position on a global reference frame to record the global coordinates of destination and target location is provided to the vehicle from the application being operated by the user. To accomplish this task, the vehicle uses GPS, with a magnetic compass and Inertial Measurement Unit (IMU) to gain information for safe traversal. The vehicle is capable to identify the non-accessible path from the paths provided on the extract optimal path information to reach a goal provided to it.

The present vehicle is battery driven, hence it offers several advantages over conventional fuel-based vehicles. This prototype is eco-friendly with almost zero emissions. It is energy efficient with low power consumption. When used for delivery services, it is comparatively cheaper in operation with low maintenance cost. In addition, it also provides assurance of reaching the destination location once properly selected.

2.0 THEORY AND LITERATURE REVIEW

The literature review section has been completely covered under six broad areas, namely, all-terrain vehicles (ATV), robot operating system (ROS), Rosbridge, GPS, ArduPilot Mega (APM) board and control systems. These are provided in the sub- sections below.

2.1 All-Terrain Vehicles

All-Terrain Vehicles (ATVs) [4] are those vehicles which can be driven on almost all surfaces such as desert areas, rugged terrains to concrete roads, grasslands, shallow watery lands etc. On a commercial scale, in the early 1970s, ATVs were introduced in markets for sale in United States with the initial models having a three-wheeler design. ATVs became more popular in the early 1980s with introduction of models having four-wheel design and thus it became a predominant choice of users worldwide [5].

2.2 Safety concerns associated with ATV

Along with significant increase in sales of ATV, increment in fatalities and injuries with usage of ATVs was also observed. Various research studies [6-11] have been conducted to test the risks of injuries associated with driving ATVs. Based on mortality and injury scores they have been compared as dangerous as motorcycles. Proper usage of protective equipment may help in reduction of most of the common injuries related to their usages. Equipment's like suitable helmet approved by Department of Transportation (DOT), protective eye-wear, gloves along with riding boots for almost all riding conditions are recommended by majority of ATV manufacturers for sports-enthusiasts or while driving on challenging terrains like aggressive riders also (such as rock crawling or hill-climbing), motocross-style protector for chest with knee/shin guards for further protection would also add up to the safety of drivers of ATV.

2.3 Classification of ATVs

According to Oregon law [12], an ATV is a two- to eight-wheeled vehicle with a motor designed for riding on unpaved surfaces. Oregon divides commercial ATVs into four classes: Class I ATV, Class II ATV, Class III ATV and Class IV ATV based on their physical structure, mechanical configuration and load carrying capacities.

International Journal of Engineering Research & Technology (IJERT)
ISSN: 2278-0181
Vol. 8 Issue 12, December-2019

2.3.1 Advantages of ATVs

ATVs offer many advantages over conventional vehicles, some of these are:

(a) Unique moving ability: ATV's are designed to locomote in all types of terrain surface conditions and the associated problems like bumps, dips, hills, etc. They are designed to give the convenience of driving in all varying terrains including areas which remain inaccessible by other vehicles, which makes them more beneficial for wide range of tasks.

(b) Strong dynamic ability: Owing to compact size and tight turning radius, an ATV has efficient capabilities of maneuvering on the narrow spaces. ATVS possess high power-to-weight ratio due to the fact that they get powered up by relatively smaller engines and also it has light weight body structure. This helps it to accelerate and brake spontaneously.

(c) Easy to repair and maintain: Being a compact system repair of mechanical components is quite simple. The mechanical simplicity of design of ATV makes maintenance and customization easier.

2.3.2 Prototype of the proposed vehicle

The proposed vehicle is designed as an all-terrain vehicle, so that it can travel on critical road surfaces and move anywhere. Thus it possesses all the advantages of ATV discussed above. Moreover, being autonomous in nature, the safety issues connected with driving this laboratory scale prototype ATV are taken into consideration through the use of advanced algorithms.

Fig 1. Mechanical Chassis with the Connected Wheels of the Designed Prototype Robot.

The developed prototype vehicle uses "Dagu Wild Thumper" 6-wheel-drive all terrain chassis. Figure 1 shows the photograph of the bare mechanical chassis used to design this prototype vehicle. This rugged chassis is designed to traverse over all types of terrains, like concrete roads, steep incline and even stairs, making it generalized research platform for any vehicle. This prototype meets the requirements of performing tasks in complex outdoor environment normally observed in a mining operations or field. The used chassis features six powerful DC motors with brass brushes and 34:1 steel gearboxes that drive large (120 mm diameter) spiked tires, and a unique "super-twist" suspension system which forms a basis to keep each wheel in contact with the ground for maximum traction while driving over uneven or bumpy or highly undulating surfaces. The suspension can be adjusted to suit different loads and road conditions. All the 6 motors associated with the six wheels can be independently controlled with the help of the motor driver L298D and a microcontroller (Arduino Mega). This provides a wider control over vehicle's speed, steering and brakes. The steering of the vehicle is done using differential motor, and the two extra wheels in the middle helps the vehicle to take turns freely, without deviating from the mean position. The chassis is equipped with two encoders (48 cpr) connected to the both the motors in the middle, which gives more accurate readings compared to the single encoder.

2.4 ROS

ROS, or Robot Operating System [13-14] is an open source framework which provides an abstraction layer for complex configurations involving both software and hardware. ROS supports several programming languages making it flexible for most of the users to resolve their problems by several means.

2.4.1 Brief History and Benefits of ROS Ecosystem

In continuation of efforts made at Stanford University amidst 2000's which involved embodied, integrated AI such as the Stanford AI Robot (STAIR) and the Personal Robots (PR) program, Willow Garage (2007) added more significant resources to extend these concepts further and create well-verified implementations. The developed software was associated with an open-source license making it a widely-used platform in the robotics research community eventually named as ROS. The ROS ecosystem now involves millions of users worldwide, working in domains varying from hobby projects to large industrial automation systems nowadays.

2.4.2 Benefits of using ROS

There are lots of benefits of usages of ROS, four of the important ones are as follows:

(a) Peer-to-peer communication: Complex robotic systems have multiple on-board computers (for performing tasks parallelly) with multiple links connected by a common network. Involving a central master for the same network involves severe congestion in a given link. Peer-to-peer communication resolves this issue efficiently. Thus, in ROS, a peer-to-peer network architecture is followed which gets coupled to a buffering system and a lookup system (a name service called 'master' in ROS), enabling each component to dialogue directly with any other, synchronously or asynchronously as required.

(b) Free and open-source: Being an open- source platform, it provides reuse of already existing functions provided by the many other ROS users. Their code is supplied in repositories as "stacks". Use of already developed robot's capabilities shared in "open- source" mode becomes relatively easy to be incrementally added to different applications with the help of ROS development environment.

(c) Thin: Development of Algorithms entangled to a lesser or greater degree with robotics OS get facilitated by this platform easily with drivers and drivers get contained in a standalone executable.

(d)Multi-language: ROS is a language-neutral platform as applications can be programmed in various languages. ROS specification generally works at the messaging layer where peer-to-peer connections get negotiated in XML-RPC, which exists in large number of programming languages.

International Journal of Engineering Research & Technology (IJERT)
ISSN: 2278-0181
Vol. 8 Issue 12, December-2019

2.4.3 Concepts of ROS

ROS gives access to a framework which allows to segregate processes, also referred to as nodes, providing a means to communicate between the nodes. Multiple nodes function together in a full robotic system in sequential and / or parallel execution modes. A node can be described as processes such as a sensor publishing data, localization algorithm, or even a data logger. Each node is dedicated to execute some particular function with larger algorithms and functionalities realized through the interactions between the nodes.

Some terminologies that have been used throughout the rest of paper have been discussed as follows:

Node - An executable unit which communicates with other nodes through message passing

Message - Unit of data exchanged between nodes

Topic - Communication channel between two or more nodes

Publisher - Node pushing data into a Topic.

Receiver - Node receiving data from a Topic Service - Remote procedure call in which Node A requests,

Node B performs some action, and Node B returns the outputs to Node A.

2.5 ROS Bridge

Rosbridge [15-16] gives us an additional level of abstraction on top of ROS and treats all of ROS as a back end. This helps developers by eradicating the need of having an intimate knowledge of low-level control interfaces, middleware build systems and execute sophisticated robotic sensing with control algorithms. The only prerequisite being understanding of middleware packages involving to build and transportation mechanisms. Rosbridge layers a simple socket serialization protocol over all of this complexity, on top of which application developers of all levels of experience can create applications.

ROS abstracts individual robot capabilities, allowing robots to be controlled through messages. It also provides facilities for starting and stopping the individual ROS nodes providing encapsulation of unified view of robot and its environment. Rosbridge also allows access to underlying ROS messages and services as serialized JavaScript Object Notation (JSON) objects, and in addition provides control over ROS node execution and environment parameters.

2.5.1 Features of Rosbridge

Rosbridge provides a JSON (JavaScript Object Notation) API (application program interface) to ROS functionality for other programs to interface with ROS. Various front end applications also interface with Rosbridge, including a WebSocket server for web browsers. Rosbridge specifically consists of protocol and its implementation. Rosbridge protocol is a specification for sending JSON based commands to ROS. This protocol is programming language and transport agnostic with the underlying idea being, any language or transport that can send JSON can talk to the Rosbridge protocol and interact with ROS.

The protocol includes publishing and subscribing to topics, making service calls, getting and setting params, and even compressing messages and more. In the second part, the Rosbridge suite package implements the Rosbridge protocol and provides WebSocket transport layer to the developer. Three of the important packages include Rosbridge library, Rosapi and Rosbridge server that are used for different applications by the developers.

2.6 ArduPilot Mega(APM)

ArduPilot Mega (APM) is a IMU system to assist autopilot that is based on the Arduino Mega platform. This system is designed to control fixed-wing aircraft, multi- rotor helicopters, traditional helicopters including ground vehicles and submarines.

The ArduPilot project [17] was started by Jordi Munoz (in 2007) and which was later released by Munoz and Anderson in 2009 (flight controller software) along with a hardware board. At present, the ArduPilot code evolution continues support for integrating and communicating with powerful companion computers for autonomous navigation, plane support for additional vertical take-off and landing (VTOL) architectures, integration with submarines, and integration with ROS platform. MAVLink extendable communication node for ROS or MAVROS, is a package that is used for the integration of APM with ROS for proper sensing, data integration and processing having facilities of proxy for ground control stations. The ArduPilot Mega (APM) comprises of three components: APM hardware board which is a physical board consisting sensors and processor, the APM firmware with code running on APM board, and the APM software, a program running on system for uploading firmware and change the settings on the APM board. The ArduPilot Mega version2.5 (APM2.5) board has been used in the prototype vehicle for this technical investigation.

2.6.1 Specifications of APM2.5 board

APM2.5 board consists of Atmel's ATMEGA2560 and ATMEGA32U-2 chips for processing and USB functions. It is an Arduino compatible device. It has 3-axis gyro, accelerometer, a high-performance barometer, a 3 axis magnetometer/digital compass (powered by Honeywell's HMC5883L-TR chip). There is an onboard 4 MP data-flash chip automatic data logger. APM2.5 supports off-board GPS and any TTL level GPS receiver can be connected to the board for obtaining the user coordinates.

2.7 Global Positioning System(GPS)

The Global Positioning System [18-20] is a based on satellites for highly accurate position, velocity and time estimation of the user location. Determination of position, time and velocity of the user, a Time of Arrival (ToA) technique is used which is based on measurement of transit time of propagation of signal from satellite to receiver. Pseudo-range between satellite and receiver is estimated by multiplication of transit time and speed of light because transit time measurement involves unknown receiver clock bias. Finally, through trilateration with the estimated pseudo ranges from several satellites, the GPS receiver can estimate user position and local time. The GPS receiver used in the prototype vehicle is Ublox M8N.

International Journal of Engineering Research & Technology (IJERT)
ISSN: 2278-0181
Vol. 8 Issue 12, December-2019

2.7.1 Specifications of Ublox M8N

The position accuracy of the used Ublox M8N GPS receiver is approximately 2 to 2.0 m circular error probability (CEP), acquisition time for the cold start is 26 s, aided start time is 2 s and the reacquisition time is 1.5 s; sensitivity to tracking and navigation is –167 dBm, to cold start is –148 dBm and that for hot start is –156 dBm, the navigation update rate is 10 Hz. The operating temperature ranges from –40 degrees C to 85 degrees C and the storage temperature lies within –40 degrees C to 85 degrees C. It features additional front-end LNA for easier integration of antenna and a front-end SAW filter increasing the jamming immunity. It is compatible with APM boards.

2.8 Control Systems

The control operations involved in tracking and following paths of an autonomous vehicle are one of the most important challenges in automation because of constraints on mobility, high-speed operation, speed of motion, complex interaction with environment and most typically lack of prior information. In general, the concepts of control system can be classified based on linearity as Linear control system and Non- linear control system. Linear control systems are those which follow the principle of superposition, which includes homogeneity and additivity. A system follows principle of superposition, if, for any two given inputs x_1 and x_2 to the system, one has y_1 and y_2 as the corresponding outputs, and then if the input $x = a. x_1 + b. x_2$ is provided to the system, then the corresponding system output should be $y = a. y_1 + b. y_2$, where a and b are constants. An example of linear control system can be a control system which consists of only resistive network and DC source. On the other hand, non-linear control systems include all those systems which do not follow the principle of superposition. A thermostat controlled heating system can be classified as non-linear control system. Another perspective for classifying the control system based on active parts may be considered as dynamic and static control system. The static control systems are those systems in which the output depends only on present input. If the main objective of control system is to maintain a physical variable at some value in presence of noise or disturbances, then static control system is used. Example of this system is a controller used to maintain the speed of generator at constant value that results in a generated voltage having frequency of 60 Hz even in presence of varying electrical power loads. The system whose output depends on present as well as past inputs is normally called dynamic system. The dynamic control system can be described as a control system in which a physical variable is required to follow, or track, some desired time function. Example of this system may be considered as a control system of robot, in which robotic chassis is made to follow some desired path in space satisfying certain criteria.

Various control methods include PID control method, LQR control method, fuzzy control method, model referenced control method etc [21-27] exists at the present time. These methods are briefly discussed below:

2.8.1 PID Controller

A proportional–integral–derivative controller (PID controller) follows a control loop feedback mechanism which finds it use in various applications requiring modulated control in a continuous manner. A PID controller calculates an error value e(t) as the difference between a desired set point (SP) and an estimated process variable (PV) for correction based on proportional, integral, and derivative terms (denoted P, I, and D respectively). The implemented control system of the prototype vehicle is based on PID control, as it is much simpler to put into practice compared to other control system algorithms, and it also provides good motion control of the vehicle.

2.8.2 LQR Controller

The Least Quadratic Regulator (LQR) controller is an optimal controller a mechanism which predicts future outputs at every time step for minimizing a global criterion/cost function. The algorithm is mainly focused on detection of controller settings which minimize the undesired deviations. The magnitude of control action gets included in cost function at each time step. This algorithm helps in reduction of work done by control systems engineer for optimization of controller. The LQR algorithm is essentially an automated way of finding an appropriate state-feedback controller. Difficulty in finding the right weighting factors limits the application of the LQR based controller synthesis.

2.8.3 Model Predictive Controller

Model predictive controllers (MPC) rely on processes having dynamic models. The main advantage of MPC is the fact that optimization of current timeslot is not allowed, keeping future timeslots in account. This mechanism is obtained by optimization of finite time horizon but implementing current time slot and performing optimization iteratively. MPC possesses the ability to take control actions by anticipating future events. This predictive ability remains absent in PID controllers. MPC follows a digital control implementation universally with achievement of faster response times through specially designed analog circuitry.

2.8.4 Fuzzy Controller

A fuzzy control system is a control system that interprets analog input values in terms of logical variables which have continuous values between 0 and 1, in contrast to classical or digital logic, possessing discrete values of either 1 or 0 (true or false, respectively).

Fuzzy controllers comprise of an input stage, a processing stage, and an output stage. The input stage performs mapping of sensors or other inputs to the corresponding truth values and membership functions. IN processing stage, appropriate rules get invoked and results get generated for each of them, after which results of all rules are combined. Finally, in the output stage, conversion of the combined results into a specific control output value takes place. Fuzzy controllers are frequently used with PID controller to make it self- tuning. Thus, instead of manually controlling the PID parameters Kp, Kd and Ki, these are adjusted through fuzzy logic controller. This controller is more adaptable to the changes, and provides

International Journal of Engineering Research & Technology (IJERT)
ISSN: 2278-0181
Vol. 8 Issue 12, December-2019

better result in comparison to conventional PID controller. However, they are more complex in implementation.

2.9 Control System used in the Developed Prototype

PID based approach has been implemented for controlling both the speed and direction of movement of the prototype robotic vehicle. Different model parameters used in this model have been discussed below:

`P` is proportional to present value of the (SP — PV), i.e., error e(t). For example, on the account of large and positive error, similar large and proportional control output is observed, taking Kp as gain factor. Usage of proportional control in a process with tasks like temperature control will result in error between set point and actual process value, as error is required to generate proportional response. If there is zero error, then there will be zero response consecutively.

`I` accounts for previous values of the (SP — PV) error for integrating them over time produce term `I`. For example, after application of proportional control portion if residual error exists, this term helps in eradicating the error by generating control outputs based on historic cumulative value of errors over previous time steps. On elimination of error, the term `Ki` ceases to grow. Thus decrement in error diminishes proportional effect which gets compensated by growth in integral effect.

`D` is a best estimate of the future trend of the difference error (SP — PV) value, based on current rate of change. This is often referred as "anticipatory control", as it effectively reduces effect of difference error by exerting control influence generated by rate of error change. Here, rapidness in change is directly proportional to dampening or control effect. The gain associated with this term is Kd.

A PID controller (1) calculates a difference or error value, e(t) as the difference between a desired set point (SP) and a measured process variable (PV) while it applies a correction based on proportional, integral, and derivative terms:

$$u(t) = Kp * e(t) + Ki \int e(t)dt + Kd * \frac{de(t)}{dt} \quad (1)$$

Here, u(t)is the process variable (like velocity) Kp is the proportional gain, Kd is differential gain and Ki is the integral gain. Since, it is not possible to realize the derivative and the integral term in the micro-controller directly, so instead of this, a discrete PID control algorithm has been used. The PID control in this vehicle is implemented in the code for the motor control module of Arduino Mega. The following pseudo-code (algorithm) shows our implementation:

```
previous_error = 0
integral = 0
loop:
    error = setpoint –measured_value
    integral = integral + error * dt
    derivative = (error – previous_error) / dt
    output = Kp * error + Ki * integral + Kd * derivative
    previous_error = error
    wait(dt)
goto loop
```

3.0 PROPOSED APPROACH

The prototype takes the inputs for its current position information as the start position from GPS module in terms of latitude and longitude. User provides the destination by clicking on the map location and the application generates the destination latitude and longitude to be used by the vehicle based on the point selected by the user. Based on the start and destination points, the vehicle obtains the accessible sequence of waypoints. Once the destination is reached, another user can take over the control of the vehicle. Algorithmic Chart for this process is shown in Figure 2.

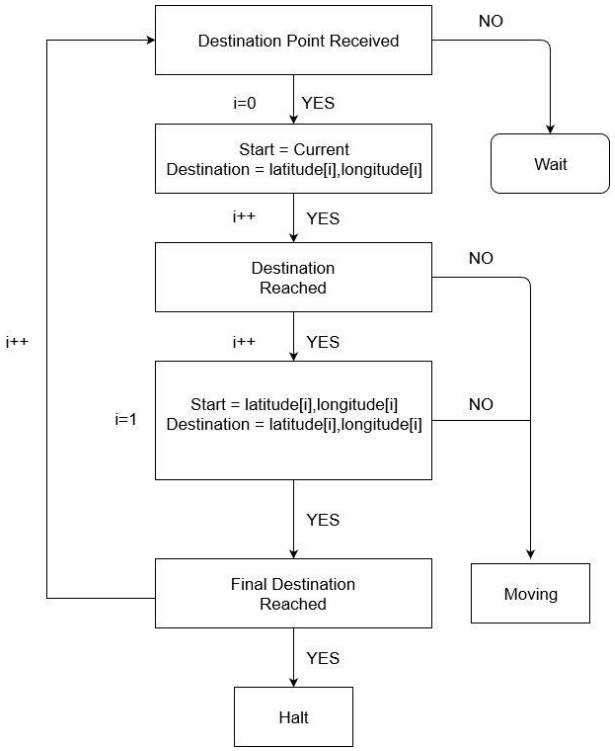

Fig 2. Algorithm Chart

Figure 3 provides the idea of traversing the path from start location to the destination point, the distance d between the start and destination points is calculated using Haversine formula [28] using equations (2), (3), and (4) given below:

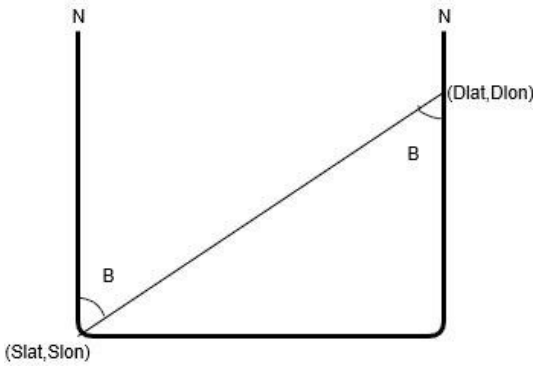

Fig 3. Distance and Bearing Angle.

$$a = sin^2\left(\frac{\Delta\psi}{2}\right) + cos\,\psi 1\,.\,cos\,\psi 2.\,sin^2\left(\frac{\Delta\lambda}{2}\right) \quad (2)$$

$$c = atan2\left(\sqrt{a}, \sqrt{1-a}\right) \quad (3)$$

International Journal of Engineering Research & Technology (IJERT)
ISSN: 2278-0181
Vol. 8 Issue 12, December-2019

$$d = R.c \quad (4)$$

Equation (5) represents the calculation for the heading (bearing) angle to be followed by the robotic vehicle.

$$B = atan2(\sin \Delta\lambda \ . \cos \psi 2 \ , \cos \psi 1 \ . \sin \psi 2 - A \quad (5)$$
$$A = \sin \psi 1 \ . \cos \psi 2 \ . \cos \Delta\lambda \quad (6)$$

Here, start latitude: $Slat = \psi 1$, *d*estination latitude: $lat = \psi 2$,start longitude:$Slon = \lambda 1$, destination longitude: $Dlon = \lambda 2$, $\Delta\lambda$ and $\Delta\psi$ are the difference in longitudes and latitudes respectively computed between the consecutive start and destination location points. R is earth's radius (mean radius = 6,371,000 m). The vehicle aligns itself according to the heading and the current angle measured from the compass, and then follows the straight line path until the distance recorded using encoder equals d. The start and destination points are updated from the way points, and the process is repeated until the final destination is reached, where the vehicles stops. Figure 4 shows the pictorial data flow of the process.

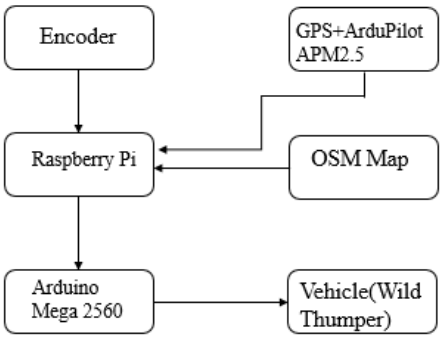

Fig 4. Sensor based Pictorial Process Flow Diagram Used for Integrated Robotic Vehicle.

4.0 EXPERIMENTAL SETUP

In this section we have described the hardware as well as the software setups for the present investigation. As of now, the hardware components are used only for positioning and localization on a global reference frame, whereas the implemented software components perform the tasks of data extraction of GPS and compass; selection of GPS coordinates from the OSM, waypoint extraction for navigation and PID controlled movement of the robotic chassis.

4.1 Hardware Setup

The prototype is built on a Wild Thumper chassis (Figure 5) having the sensors attached on the required locations in the different components. Each of the six motors are mounted independently on six different suspension units attached with the chassis. All the six wheels are separately attached with the six motors fitted with the chassis in such a way that they remain protruded from the chassis.

Three independently working controlling units are placed in the lower casing, where two motors, placed on the left and right sides of the chassis, can be controlled and driven from a single motor controlling unit. Hence, all the six wheels can now be driven and controlled by the motor controllers. There are encoders placed in the middle two wheels of the robotic vehicle as per the specifications provided earlier. In addition, a Raspberry-Pi3 is attached on the robotic vehicle for processing purposes.

The sensors including GPS module, digital compass on APM2.5 board and the APM2.5 board itself are placed and connected with the processing unit. The navigation code and implementation of PID controller logic for the robotic vehicle is incorporated in an Arduino Mega2560 board placed following the RaspberryPi3 board (as shown in Figures 4 and 5). Since this prototype has to cover distances far away from AC power sources, it is fitted with a pack of 3 celled Li-Po battery having a maximum rated voltage of 11.4V and capacity of 5200mAh.

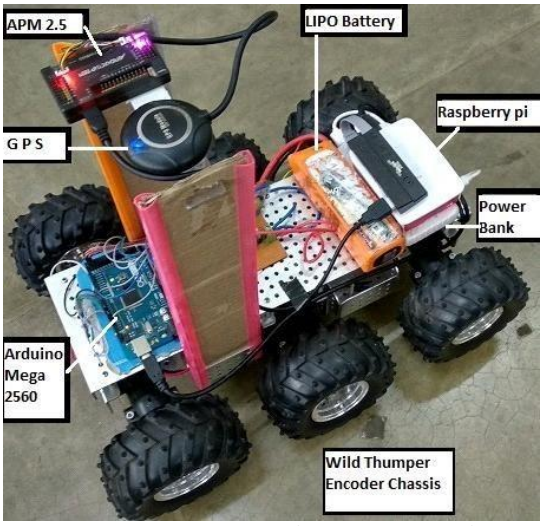

Fig. 5. Hardware Setup with Labels of Each Component of Our Study

4.2 Software Setup

The software setup consists of four distinct modules for (a) data extraction of GPS and compass using ROS nodes, (b) destination selection, and (c) waypoint extraction. These are explained below in brief:

(a) Extracting GPS and Compass Data Two of the ROS packages are used for extracting GPS and Compass data. These are, ArduPilot and Mavros. The first one burns the code / firmware on the APM2.5 board whereas the second one, as described earlier, is used for providing proxy for the ground control stations. Figures 6 and 7 display the sample compass data and GPS data. The bearing / heading angle has been shown to be extracted from the compass whereas of all the extracted data points from GPS, we use the latitude and longitude values for our navigation purposes. The variation in height is neglected for this study.

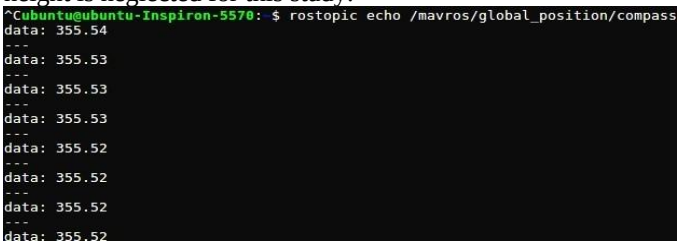

Fig.6 Sample Compass Data

International Journal of Engineering Research & Technology (IJERT)
ISSN: 2278-0181
Vol. 8 Issue 12, December-2019

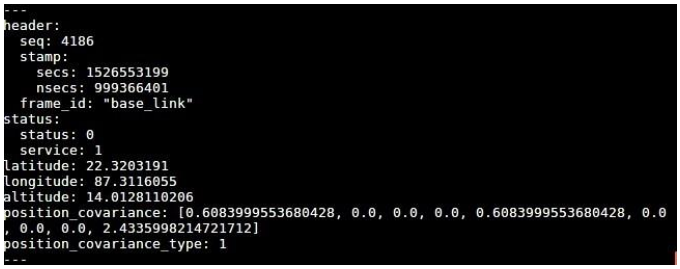

Fig.7. Sample GPS Data

(b) Destination Selection: The destination point is provided to the vehicle by the user from ROS-OSM application [31-33] The Rosbridge provides ROS functionalities for non-ROS users like web browsers. Here, Rosbridge connects the OSM application to the ROS, such that the GPS and compass data which is being published on their respective topics can be known to the application for locating the present position and at the same time passing the selected information to the ROS subscriber node. Figure 8 represents the graphical user interface where the start and destination. Location points are displayed. The destination point is selected by the user on the same GUI before the same gets displayed. With every movement of the robot, the current location gets refreshed on the GUI and upon reaching the destination suitable message gets displayed to the user.

Fig. 8. GUI of OSM Application for Start and Destination points showing the current and intended positions respectively. (The dotted line represents the path followed by robot)

(c) Waypoint Extraction: A python script subscribes to the start and destination points publisher node, provided by the OSM application and publishes the driving-accessible waypoints with the help of "Google Maps" module. These waypoints are published in the form of python list which can be subscribed by the main node.

During the experiment, based on the destination selected by user on the map (as shown in Figure 6), the following results are obtained Figure 9 shows the execution of the module starting from the selection of the destination point by the user (represented by a TRUE value for goTo parameter), displaying the destination latitude, destination longitude, start latitude and start longitude. Once this information is made available, the nearest locations points are obtained from the "Google Maps" module. Also, the drivable shortest path, from among the selected paths, is selected. The available list of waypoints along this selected path is then made available to the python list for the robot to move.

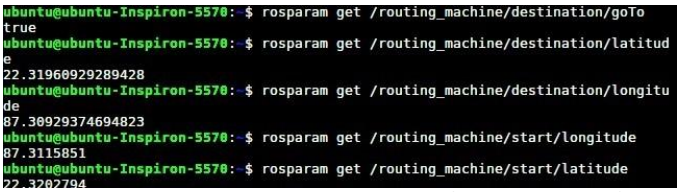

Fig 9. Start and Destination point data as obtained from the OSM interface using 10 GPS Satellites

The waypoints obtained are all those accessible points for driving between the start and destination position. Figure 10 shows a sample of the obtained drivable waypoints longitude and latitude data for one of the experiments conducted in this study. The topics containing these data are /latitude (depicting a ROS topic) and /longitude (depicting another ROS topic). These topics along with other topics containing information like distance travelled by the vehicle subscribed by the main node, here Figure 11. The main node keeps track of the current distance travelled by the vehicle and the current compass angle and compares it with the desired distance and angle. Based on these comparisons, control action is provided to the vehicle.

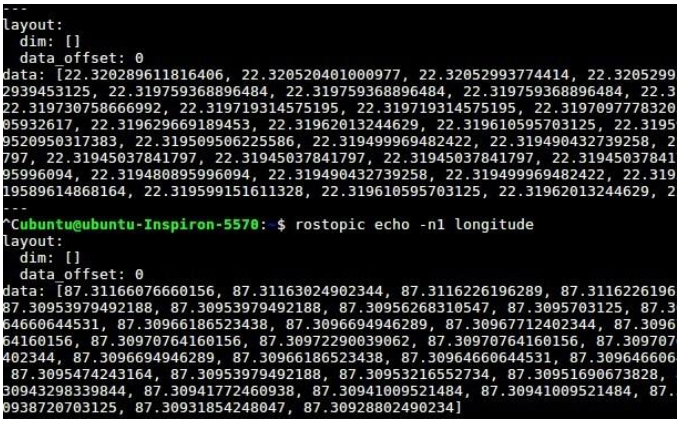

Fig. 10. Extracted driving waypoints from Google Maps modules with python scripts, displayed as list of latitudes and longitudes between the selected source and destination points

5.0 ANALYSIS

The prototype of the proposed vehicle was tested in the practical environment. The vehicle worked properly, and the results from the experiment [Figure 6-11] show that all the commands were tested successfully, and that vehicle followed the expected path.

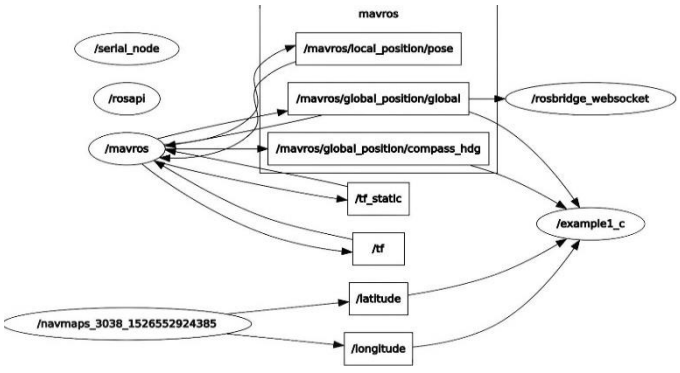

Fig. 11. RQT graph of all active nodes and topics for executing the robot movements in this study.

International Journal of Engineering Research & Technology (IJERT)
ISSN: 2278-0181
Vol. 8 Issue 12, December-2019

However, due to dependency on the GPS, the accuracy in positioning of the vehicle is limited to that of GPS module itself. It depends on the strength of signal received, noise, no. of logged satellites. The variation in no. of logged satellites is due to the GPS ephemeris, different weather conditions, places, clear or cloudy sky etc. Beside GPS, the orientation of the vehicle depends on the data received from compass. Due to variation in the local magnetic field strength, EMI interference and noise, the compass data is also subjected to errors. So, both the GPS receiver data and the compass data were analyzed to find its accuracy and precession.

No. of Visible Satellites	Mean Latitude (degrees)	Mean Longitude (degrees)	Position Error (m)	Position Error Variance	RMSE (m)
4	22.320548616	87.3115367420	9.53168	11.46502	3.38600
5	22.320570861	87.3115160680	7.24755	8.86195	2.97690
6	22.320609283	87.3115381658	6.92668	5.99196	2.44785
7	22.320616776	87.3115022353	4.74058	3.92244	1.98051
8	22.320620645	87.3115384750	4.25373	3.46902	1.86253
9	22.320625071	87.3115628390	2.59038	2.98858	1.72875
10	22.320638580	87.3115276150	1.63441	1.25482	1.12019
11	22.320633490	87.3115267540	0.97475	0.24492	0.49489
12	22.320627257	87.3115217710	0.24397	0.01245	0.11158
13	22.320625535	87.3115250870	0.31703	0.00585	0.07651
14	22.320626292	87.3115221280	0.06895	0.00136	0.03693

Fig. 12. Table showing the mean of data recorded by GPS when it is kept at fixed position with respect to change in no. of visible satellites

The GPS receiver was kept at a fixed position, and sample data were obtained, for no. of visible satellites ranging from 4 to 14. The mean position error was recorded, along with the variance and root mean square error (RMSE). It was observed that the accuracy and precision of the receiver increased significantly with increase in no. of visible or logged satellites. Figure 12 shows the mean position (latitude and longitude), along with position error, variance and RMSE with respect to no. of logged / visible satellites. The graph in Figure 13 shows the error quantities with different no. of satellites and it can be interpreted that the corresponding accuracy and precision of the GPS readings increase with the increase in no. of visible / logged satellites. The average position accuracy is observed to be 3.5 m with the average variance and RMSE of 3.47 and 1.86 m respectively. Along with the GPS data, the compass data was also analyzed. It was observed that when the vehicle was kept static in one direction, and recorded data were analyzed, the mean bearing angle came to be 4.0325 degree NE, the variance was of 0.0010, and the root mean square error (RMSE) value was 0.03083 degree. This shows that the compass has very low variation in the reading in static mode of operations, while in the dynamic mode of operations RMSE has been computed to be varying from0.14 to 2.66 degrees.

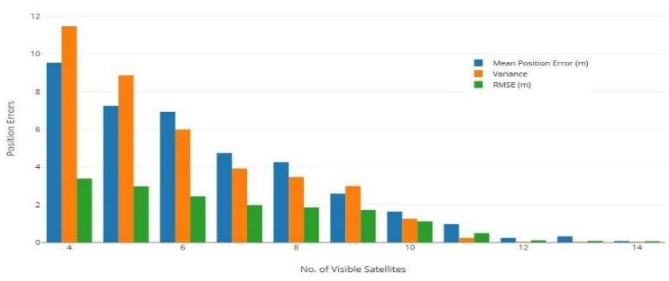

Fig13. Bar Graph showing errors in GPS data vs. No. of visible satellites when the GPS receiver is static.

6.0 CONCLUSION

In this paper, all terrain robot movement using global mapping architecture was implemented with the help of a designed prototype vehicle. The goal was to enable the user to give the vehicle the desired destination point using a global map remotely, and it must reach that point following the shortest path drivable available from extracted GPS waypoints. Experiments were conducted on our prototype, and it successfully reached the destination each time, which was provided remotely using the OSM map. The results from the experiment show that the ROS and the Rosbridge application were successfully integrated with the vehicle which was used for vehicle control from remote locations. The all-terrain movement capability of the vehicle gives it an extra edge over other vehicles. The data from GPS receiver and compass were recorded for analysis of the accuracy and precision measures.
Thus, it may be concluded that the use of the proposed vehicle in the real world scenario is practically feasible, and economical. Using this vehicle, the delivery and transportation needs can be easily met by the users at low cost, along with all the benefits that has already been discussed in this paper.
The only requirement of the vehicle is that both user and vehicle must be in the same network which is its limitation. This is being planned to be remove in future work, by providing mechanism to control the vehicle from any network, and from any part of the globe. Also, the focus was mainly on navigation part for this study, so collision detection-avoidance and vision-based interpretation capabilities have not been implemented, but these will be incorporated to enhance the working of the vehicle in the future work.

7.0 REFRENCES

[1] R. Dhanasingaraja, S. Kalaimagal, and G. Muralidharan, “Autonomous vehicle navigation and mapping system,” in Division of Mechatronics, 2014 International Conference on Innovations in Engineering and Technology.,3(3) (2014).
[2] I. Shimchik, A. Sagitov, I. Afanasyev, et al., “Golf cart prototype development and navigation simulation using ros and gazebo,” in MATEC Web of Conferences, 75, 09005, EDP Sciences (2016)
[3] S. Panzieri, F. Pascucci, and G. Ulivi, “An outdoor navigation system using gps and inertial platform,” IEEE/ASME transactions on Mechatronics 7(2), 134–142 (2002).
[4] R. Jarvis, “An all-terrain intelligent autonomous vehicle with sensor-fusion-based navigation capabilities,” Control Engineering Practice 4(4), 481–486 (1996).
[5] U. C. P. S. Commission et al., “Standards for all terrain vehicles and ban of three-wheeled all terrain vehicles; notice of proposed rulemaking,” Federal Register 71(154), 45904–45962 (2006).
[6] G. B. Rodgers, “The effectiveness of helmets in reducing all-terrain vehicle injuries and deaths,” Accident Analysis & Prevention 22(1), 47–58 (1990).
[7] M. E. Aitken, C. Graham, J. B. Killingsworth, et al., “All-terrain vehicle injury in children: strategies for prevention,” Injury Prevention 10(5), 303–307 (2004).
[8] G. B. Rodgers and P. Adler, “Risk factors for all-terrain vehicle injuries: a national case-control study,” American Journal of Epidemiology 153(11), 1112–1118 (2001).
[9] N. Z. Cvijanovich, L. J. Cook, N. C. Mann, et al., “A population-based assessment of pedi-atric all-terrain vehicle injuries,” Pediatrics 108(3), 631–635 (2001).
[10] J. A. Acosta and P. Rodr´ıguez, “Morbidity associated with four-wheel all-terrain vehicles and comparison with that of motorcycles,” Journal of Trauma and Acute Care Surgery 55(2), 282–284 (2003).
[11] A. Fonseca, M. G. Ochsner, W. Bromberg, et al., “All-terrain vehicle injuries: are they dan-gerous? a 6-year experience at a level i trauma

International Journal of Engineering Research & Technology (IJERT)
ISSN: 2278-0181
Vol. 8 Issue 12, December-2019

center after legislative regulations expired," The American surgeon 71(11), 937–941 (2005).
[12] "Oregon ATV Laws and Rules." The ROS Wiki, https://www.oregon.gov/oprd/ ATV/docs/hb_laws_rules.pdf. (Accessed: May 22, 2018).
[13] M. Quigley, K. Conley, B. Gerkey, et al., "Ros: an open-source robot operating system," in ICRA workshop on open source software, 3(3.2), 5, Kobe, Japan (2009)
[14] "Robot Operating System." In Wikipedia, The FreeEncyclopediahttps://en.wikipedia.org/w/index.php?title=Robot_ Operating_System& oldid=845261072. (Accessed: May 20, 2018).
[15] C. Crick, G. Jay, S. Osentoski, et al., Rosbridge: ROS for Non-ROS Users, 493–504. Springer International Publishing, Cham (2017).
[16] C. Crick, G. Jay, S. Osentoski, et al., "Ros and rosbridge: Roboticists out of the loop," in Proceedings of the seventh annual ACM/IEEE international conference on Human-Robot Interaction, 493–494, ACM (2012).
[17] "ArduPilot." In Wikipedia, The Free Encyclopedia https://en.wikipedia.org/w/index.php?title=ArduPilot&oldid=845034 896. (Accessed: May 20, 2018).
[18] J. Parthasarathy, "Positioning and navigation system using gps," International Archives of the Photogrammetry, Remote Sensing and Spatial Information Science 36(Part 6), 208–212 (2006).
[19] S. Van der Spek, J. Van Schaick, P. De Bois, et al., "Sensing human activity: Gps tracking," Sensors 9(4), 3033–3055 (2009).
[20] W. Shi, Q.-J. Kong, and Y. Liu, "A gps/gis integrated system for urban traffic flow analysis," in 2008 11th International IEEE Conference on Intelligent Transportation Systems, 844–849, IEEE (2008).
[21] Y. Li, K. H. Ang, and G. C. Chong, "Pid control system analysis and design," IEEE Control Systems Magazine 26(1), 32–41 (2006).
[22] P. Zhao, J. Chen, Y. Song, et al., "Design of a control system for an autonomous vehicle based on adaptive-pid," International Journal of Advanced Robotic Systems 9(2), 44 (2012).
[23] C. E. Garcia, D. M. Prett, and M. Morari, "Model predictive control: theory and practice—a survey," Automatica 25(3), 335–348 (1989).
[24] Z.-W. Woo, H.-Y. Chung, and J.-J. Lin, "A pid type fuzzy controller with self-tuning scaling factors," Fuzzy sets and systems 115(2), 321–326 (2000).
[25] E. Natsheh and K. A. Buragga, "Comparison between conventional and fuzzy logic pid con-trollers for controlling dc motors," International Journal of Computer Science Issues (IJCSI) 7(5), 128 (2010).
[26] S. Bouabdallah, A. Noth, and R. Siegwart, "Pid vs lq control techniques applied to an indoor micro quadrotor," in 2004 IEEE/RSJ International Conference on Intelligent Robots and Sys-tems (IROS)(IEEE Cat. No. 04CH37566), 3, 2451–2456, IEEE (2004).
[27] S. Puntunan and M. Parnichkun, "Online self-tuning precompensation for a pid heading con-trol of a flying robot," International Journal of Advanced Robotic Systems 3(4), 44 (2006).
[28] C. Veness, "Movable type scripts: calculate distance, bearing and more between lati-tude/longitude points," online], Movable Type Scripts, nd. Available at: http://www. Mov-abletype. co. uk/scripts/latlong. html [Accessed: 10 Dec. 2016] (2012).
[29] "ROS/Network Setup." The ROS Wiki, http://wiki.ros.org/ROS/ NetworkSetup. (Accessed: May 22, 2018)
[30] S. S. H. Hajjaj and K. S. M. Sahari, "Establishing remote networks for ros applications via port forwarding: A detailed tutorial," International Journal of Advanced Robotic Systems 14(3), 1729881417703355 (2017).