AUTONOMOUS NAVIGATION AND CONTROL OF DIFFERENTIAL DRIVE ROBOT VEHICLE

A project report submitted for the partial fulfilment of the Bachelor of Technology Degree in Electrical Engineering under Maulana Abul Kalam Azad University of Technology

BY

ARNAB MONDAL
(ROLL NO:10401616124 REGISTRATION NO:161040110622)

AVINASH KUMAR
(ROLL NO:10401616121 REGISTRATION NO:161040110625)

INDRANIL CHANDRA
(ROLL NO: 10401616107 REGISTRATION NO: 161040110639)

KAUSANI ROY
(ROLL NO: 10401616106 REGISTRATION NO: 161040110640)

MD AZHAR FIROZ
(ROLL NO: 10401616101 REGISTRATION NO: 161040110645)

NIDHI SUMAN
(ROLL NO: 10401616093 REGISTRATION NO: 161040110653)

Under the Guidance of:
DR. PROF. MADHUMITA PAL

Department of Electrical Engineering
For the Academic Year 2019 - 20

IEM
श्रद्धावान लभते ज्ञानम्
INSTITUTE OF ENGINEERING & MANAGEMENT
Good Education, Good Jobs

Institute of Engineering & Management
Y-12, Salt Lake, Sector-V, Kolkata-700091
Affiliated To:

MAKAUT WB
Utech
Maulana Abul Kalam Azad University of Technology, West Bengal
formerly known as West Bengal University of Technology
In Pursuit of Knowledge and Excellence

Maulana Abul Kalam Azad University of Technology
BF-142, Salt Lake, Sector I, Kolkata-700064

CERTIFICATE
<u>TO WHOM IT MAY CONCERN</u>

This is to certify that the project report entitled "**AUTONOMOUS NAVIGATION AND CONTROL OF DIFFERENTIAL DRIVE ROBOT VEHICLE**", submitted by

1. ARNAB MONDAL
(*Registration No.* 161040110622 *of Electrical Department Roll no. 10401616124)*,

2. AVINASH KUMAR
(*Registration No.* 161040110625 *of Electrical Department Roll no. 10401616121)*,

3. INDRANIL CHANDRA
(*Registration No.* 161040110639 *of Electrical Department Roll no. 10401616107)*,

4. KAUSANI ROY
(*Registration No.* 161040110640 *of Electrical Department Roll no. 10401616106)*,

5. MD AZHAR FIROZ
(*Registration No.* 161040110645 *of Electrical Department Roll no. 10401616101)*,

6. NIDHI SUMAN
(*Registration No.* 161040110653 *of Electrical Department Roll no. 10401616093)*,

Students of **INSTITUTE OF ENGINEERING & MANAGEMENT,** in partial fulfilment of requirements for the award of the degree of **Bachelor of Technology in Electrical Engineering,** is a bona fide work carried out under the supervision and guidance of **Dr. Prof. Madhumita Pal mentor** during the final year of the academic session of 2016-2020.The content of this report has not been submitted to any other University or Institute for the award of any other degree.

It is further certified that work is entirely original and its performance has been found to be quite satisfactory.

Madhumita Pal.

Dr. Prof Madhumita Pal
Project Guide

Dept. of Electrical Engineering
Institute of Engineering & Management

Prof. Tapas Kumar Datta
H.O.D

Dept. of Electrical Engineering Institute of Engineering &Management

Prof. Dr.A.K. Nayak
Principal
Institute of Engineering & Management
Sector-V, Salt Lake Electronics Complex, Kolkata-700091

ACKNOWLEDGEMENT

We should like to take this opportunity to extend our gratitude to the following revered persons without whose immense support, completion of this project wouldn't have been possible.

We are sincerely grateful to our advisor and mentor **Dr. Prof. Madhumita Pal** of the Electrical Engineering department, IEM Kolkata, for his/her constant support, significant insights and for generating in us a profound interest for this subject that kept us motivated during the entire duration of this project.

We would also like to express our sincere gratitude to **Prof. Dr. Satyajit Chakrabarti** (Director, IEM)**,** **Prof. Dr. Amlan Kusum Nayak (**Principle, IEM) and **Prof. Tapas Kumar Datta**, HOD of **Electrical Engineering Department** and other faculties of Institute of Engineering & Management, for their assistance and encouragement.

Last but not the least, we would like to extend our warm regards to our families and peers who have kept supporting us and always had faith in our work.

Arnab Mondal
Reg. No: 161040110622
Dept. of Electrical Engineering
Institute of Engineering & Management, Kolkata

Avinash Kumar
Reg. No: 161040110625
Dept. of Electrical Engineering
Institute of Engineering & Management, Kolkata

Indranil Chandra
Reg. No: 161040110639
Dept. of Electrical Engineering
Institute of Engineering & Management, Kolkata

Kausani Roy
Reg. No: 161040110640
Dept. of Electrical Engineering
Institute of Engineering & Management, Kolkata

Md Azhar Firoz
Reg. No: 161040110645
Dept. of Electrical Engineering
Institute of Engineering & Management, Kolkata

Nidhi Suman
Reg. No: 161040110653
Dept. of Electrical Engineering
Institute of Engineering & Management, Kolkata

ABSTRACT

Robotics has been a dominant contributor to the development of human society over the years. The purpose of this project is to develop a control strategy for a differential drive autonomous mobile robot. Autonomous navigation and control of the mobile robot is a priority in this project. A navigation system is a program that provides graphical maps, co-ordinates or directions to a destination. The computation of a shortest path between different locations appears to be a key problem in the road networks.

Even now the problem still persists to find the shortest path on road networks. The wide range of applications was introduced to overcome the problem by developing various shortest path algorithms. Dijkstra's algorithm is one of them which is used in this project to find the shortest path from one node to another node. Simulation results are carried out to find and track for the shortest path by using Dijkstra's algorithm. Also, Simulation results are presented to verify the effectiveness of the proposed navigation system. The robot can reach to its final point without any error in distance and direction.
.

TABLE OF CONTENTS

1. INTRODUCTION: -

Now we are in the fourth industrial revolution of the world where world technology is going towards the cyber physical system,Internet of things,networking,automation etc.and the main trends of new technology towards the robotics and industrial automation.where robotics technology plays a important role in several sectors like factories, laboratories, warehouses, energy plants, hospitals, and many other industries,Robots are used for assembling products, handling dangerous materials, spray-painting, cutting and polishing, inspection of products. The number of robots used in tasks as diverse as cleaning sewers, detecting bombs and performing intricate surgery is increasing steadily, and will continue to grow in coming years.As such, the robot interacts with many tools and other types of equipment and therefore, must model its environment,control its motion and identify objects by using the navigation system and manage assigned tasks with its control system. A robot's navigation system controls three functions in real-time: (1) path planning, (2) self-localization and (3) motion control. The first is the process of finding an optimal path for movement from the start point to the destination while avoiding obstacles. The second is the robot's ability to determine its position and orientation within its environment. The third is the essential task of enabling the robot to operate in its environment. The assigned tasks enable mobile robots to perform specific useful duties within its environment, such as grasping and relocating objects .And the Internet of things gives the extended path of using this robotics technology,it transformed existing communications between machines to machine, machines to human in great extent and helped word to move towards more connected systems. This connected system will help all stakeholders to generate process and share the gathered data from various sources/devices located across variety of platforms.

1.1 PROBLEM STATEMENT:-

Despite of the attempts and the research work by the robot researchers to emulate human intelligence and appearance, the result is not achieved. Most robots still cannot see and are not versatile object is not properly recognized by it. For the effective and proper mechanism of robotics technology it is important to prioritize the inefficiency associated in it, generally the autonomous robots have been considered for differnt service applications,like weight lifting,the abandoned places such as place contaminated with radiations, gases or hazardous substances etc.,without any human intervention not only this it has very extended application in industries also Service robots can include intelligent wheelchairs or vacuum cleaners and medicine or food delivery robots. However, for search services, robots must recognize specific objects, which they may then be required to approach, grasp and relocate. Novel challenges exist in developing a control system that helps a mobile robot to navigate and search its environment. These include constructing an optimal navigation system that enables the mobile robot to search the entire area, because the target might have an equal probability of being at any location. As the robot performs a visual search, the choice of an exploration strategy and vision-based object recognition technique is difficult. The search strategy that directs the robot to move to a search site and to relocate as required, involves selecting the vision sensor's viewpoint. One aspect is the object detection process, which is so challenging because the robot needs to navigate and place the object in the field of view. In a classical path planning process, the robot is aware of the start and target locations. However, in a search robot application, the target position is unknown and therefore, the exploration path should cover the entire area and maximize the probability of detecting the target object. Lastly, the robot needs to manipulate the detected object by implementing specific task code. This technology is yet to capture new more areas of its operation and will be very beneficial in its progress to the society as well. So, it is necessary to put an insight to the robotic autonomous navigation technology as it has the huge application and welfare is yet to be applied and explored fully. The purpose of this overview is to provide the present and future trends of robotics.

1.2 RESEARCH QUESTIONS: -

- The problem statement identified the challenges in constructing a navigation system for an indoor search robot. Part of this research involves designing systems that are capable of overcoming the challenges. The general question is: How should a self-navigating mobile robot control system be designed?
- Another aspect of the research is: How should the different types of sensors be integrated within the control system for the search robot?
- The accuracy of a measurement system will dramatically rise if the robot is equipped with the high number of sensors. However, this increases the robot's price and leads to a more complex control system in its implementation. Therefore, the number of sensors must be reduced without affecting the efficiency of the robot motion. The question is: How should a minimum number of sensors be attached on the robot's body for autonomous navigation?
- Although there are many options, another important question is: How should vision technology be integrated with robot technology for autonomous navigation?
- Which algorithms should a mobile robot use to search and locate objects in the visual field?
- How can a robot be enabled to perform its tasks in different terrains? (The research has the potential to identify limitations on the navigation system and path planning methods, due to terrain.)
- Which custom-built instruments are needed for the robot's navigation system, which is based on the vision system and range sensors, to optimally function in the intended simulation of an industrial environment?
- The issue of scalability with regard to the size of the robot must be addressed, as in general, this issue has not received sufficient attention by researchers. Thus, this poses the question: Is the legged robot used in this research scalable for use in industrial tasks?
- Whilst no large-scale robots will be designed in this project, the question to be addressed is: What are the theoretical challenges in scaling the model robots used to a size useful to industry?

1.3 AIMS AND SIGNIFICANCE: -

This research work will be utilizing the power of IoT mechanism based on raspberry pi framework to monitor the abandoned places such as place contaminated with radiations, gases or hazardous substances etc.,without any human intervention not only this it has very extended application in industries also. The proposed research work provides mechanism of rover which is able to monitor/survey the place with self-navigation mechanism so here by using GPS module on the rover which helps it to mark its presences on the map and the enviornment which will be known by us and here we can easily set the starting and destination point and the rover can autonomously reached it's target location and it's differential drive mechanism can helps it to avoid the obstacle with using of ultrasonic sensor.And this all task can be performed together by using respberry pi which is to be pre-programed by us.And here we are using the technique of programe in PYTHON which is Path-planning algorithm can be considered as the process of navigating a robot around a configured space, which has a number of obstacles in it that have to be avoided. The problem to find a "shortest" path from one vertex to another through a connected graph is of interest in multiple domains .One of the earliest and simplest algorithms is Dijkstra algorithm, which we are using here, Starting from the initial vertex where the path should start, the algorithm marks all direct neighbors of the initial vertex with the cost to get there. It then proceeds from the vertex with the *lowest* cost to all of its adjacent vertices and marks them with the cost to get to them via itself *if* this cost is lower., this way it can proceed further point and terminated at target point.This research work will be useful to monitor the places from unauthorized persons in public and private sectors.

1.4 METHODOLOGY: -

Avoiding the obstacles in unstructured environment is itself a very challenging task and it was achieved with the help of multiple sensors or fusion of sensor data.

GPS sensor is used to move the vehicle from source to destination wherein the image processing is used to detect and avoid the obstacles.

To detect the obstacle with edge detection technique which stops the robot in front of obstacle was itself very challenging, similarly the accuracy of detection of obstacles
should also be increased.

Development of low cost robot to the transport equipment's form one place to another place controlled by using single board computer . This robot is a machine capable to perform the task autonomously or semi-autonomously which is guided by the controllers.
Many industries are using robots to perform the task with higher degree of precision. Accordingly, in light of the requirement and purpose of usage the task of sensor integration is achieved. In this research work the proposed rover is capable of detects and avoids obstacles in the travel path of rover under dynamic environment. The task of robot control system is divided into three parts. First, the exploration or navigation task includes finding a suitable exploration path that covers the entire environment with

minimal sensing requirements and then constructing this path for the robots. Second, the visual-search task involves finding appropriate image processing algorithms that are suitable for object detection and then implementing and assessing them with the robots. This also includes rotating and then choosing a suitable viewpoint field for the camera. Third, the relocating task consists of using a robot’s gripper to grasp the detected desired object and move it to the assigned final location. The relocation process also involves approaching both the target and the delivery site, using a visual tracking technique.

To understand the obstacles the ultrasonic sensors, camera module were integrated on
raspberry pi board. A path planning Algorithm is used to find out path between source to destination by using GPS based navigation. Miniature robot having poor navigation ability because of limited perception ability in the constrained space to avoid the obstacles in an unfamiliar environment.

2. BACKGROUND AND LITERATURE REVIEW

2.1 BACKGROUND

The robot is a machine capable to perform task autonomously or semi-autonomously which is guided by the micro-controllers. Many industries are using robots to perform the task with higher degree of precision. Accordingly, in light of the requirement and purpose of usage the task of sensor integration is achieved. The first industrial robot was developed and used in industry by General Motors in 1961. Since then industrial robots have been widely used in manufacturing settings for performing various tasks; especially repetitive, heavy and hazardous processes. Mostly, industrial robots are fixed and designed to work within a limited operating range. More recently, mobility has been added to industrial robots, which means the robot can perform the same tasks in different locations. Not surprisingly, 'The World Robotics Report produced by the United Nations Economic Commission for Europe predicts massive growth in the robot industry over the next decade'.

2.1.1 MOBILE ROBOT MECHANISM

Mobile robots can be classified into three categories depending on their ground locomotion configuration: wheeled, legged and articulated. Each type includes specific characteristics that make them appropriate for particular classes of applications. Like, wheeled robots use rotational devices in their motion, such as wheels and tracks. Their advantage is that they usually have simple mechanisms, a low total weight and are fast and efficient when they move on structured, regular surfaces. Therefore, they are utilized in almost all industrial applications. But they are inefficient on very soft or rough surfaces, such as outdoor, unpaved terrains also the wheeled robot consumes high energy when it wants to move on an uneven surface or over a small obstacle.

2.1.2 MOBILITY AND AUTONOMY

Mobility is the term which signifies the ability of robots to move freely from one location to another in an environment to perform their tasks. If the movement is controlled remotely by an operator, the robot is called non-autonomous. On the other hand, the autonomous robot assesses its environment by using various sensors without being operated.

2.1.3 ROBOT NAVIGATION

Robot navigation is the ability of the autonomous mobile robot to plan its motion in real-time and to navigate safely from one place to another. The robust navigation process requires three aspects, namely: path planning, self-localization and motion control.

- Path planning is the process of finding an optimal path from a start point to the target location without any collisions.
- Localization means that the robot estimates its position relative to specific objects within the environment.
- Motion control is the robot's ability to transfer sensory information into accurate physical movement in a realistic world. The process of robot navigation is a complex, technological problem as it

determines a robot's autonomy and reliability in performing assigned tasks; it has been widely researched since the 1970s.

2.1.4 EXAMPLES OF APPLICATION AREAS

Typically, mobile robots are developed to replace human beings in hazardous work or relatively inaccessible situations, such as: exploration of nuclear power plants, undersea areas and space missions. Another potential application is search and rescue for lost or injured people where the robot must explore the entire searched area. Also use of these Robot include security robots, Gas detection sensor, Temperature sensor, humidity sensor, Google autonomous car, autonomous vacuum cleaners and lawn mowers, toxic cleansing, tour guiding, personal assistants to humans, etc. Even in old times in Industries there were a lot of work that were only done by humans with life risk which this bot can replace now. Even the Traffic System modulation is a very effective use of these robots.

2.2 LITERATURE REVIEW

The most important three aspects required of a mobile search robot are: navigation (exploration path), target finding and control of a vision sensor.

The former is carefully planned to cover the robot's entire environment while taking account of the visibility of the target and optimizing both navigation time and collision avoidance. The navigation system must help the robot approach and observe the target efficiently through optimal object recognition techniques; typically using vision sensors supported by image processing techniques. The control of the vision sensor includes selection of the camera's viewpoint.

2.2.1 NAVIGATION: THE EXPLORATION PATH

A search robot navigates in an environment that typically has a starting point, a target object and a number of obstacles of random shapes and sizes. As such, the starting point is known whereas the target position is unknown. The robot moves from the starting point with the objective of finding the target. The robot must find an obstacle-free, continuous path that covers the entire environment. It should also localize itself within the environment and be aware when the search process is accomplished.

2.2.1.1 NAVIGATION STRATEGY

Navigation strategies differ depending on whether the environment is static (static obstacles) or dynamic (static and dynamic obstacles). Both categories can be subdivided into unknown and known environments. In the latter, information is provided on the location of obstacles before motion commences. Across the various environments, there are many navigation algorithms that address the robot navigation problem. All navigation planning algorithms assume that the mobile robot has detailed knowledge of the start and target locations and thus, of the direction between them, so that it can find an optimal path between these two locations and avoid obstacles. All the Navigation

Algorithms used in path planning program are offline Path planning Algorithms given that the obstacle and destination coordinates are given. The following are the Algorithms that are used in path planning are listed below:

2.2.1.1.1 Dynamic Window Approach

2D Path Planning can be done using various ways and one way is obviously Dynamic Window Approach which is a very well-known method which have been used in this domain and used nowadays. Unlike other avoidance methods, the dynamic window approach is derived directly from the dynamics of the robot, and is especially designed to deal with the constraints imposed by limited velocities and accelerations of the robot. It consists of two main components, first generating a valid search space, and second selecting an optimal solution in the search space. The search space is restricted to safe circular trajectories that can be reached within a short time interval and are free from collisions. The optimization goal is to select a heading and velocity that brings the robot to the goal with the maximum clearance from any obstacle.

2.2.1.1.2 GRID BASED SEARCH

a. DIJKSTRA ALGORITHM

This Algorithm s an algorithm for finding the shortest paths between nodes in a graph, which may represent, for example, road networks. The algorithm exists in many variants. Dijkstra original algorithm found the shortest path between two given nodes, but a more common variant fixes a single node as the source node and finds shortest paths from the source to all other nodes in the graph, producing a shortest-path tree.

b. A STAR ALGORITHM

A* is a graph traversal and path search algorithm, which is often used in computer science due to its completeness, optimality, and optimal efficiency. One major practical drawback is its space complexity, as it stores all generated nodes in memory. Thus, in practical travel-routing systems, it is generally outperformed by algorithms which can pre-process the graph to attain better performance, as well as memory-bounded approaches; however, A* is still the best solution in many cases.

c. POTENTIAL FIELD ALGORITHM

A potential field algorithm uses the artificial potential field to regulate a robot around in a certain space. For our ease, we consider a space to be divided into a grid of cells with obstacles and a goal node. Potential Fields generated in a 2D Space.

d. BELLMAN FORD ALGORITHM

The Bellman–Ford algorithm computes single-source shortest paths in a weighted digraph. It uses the same concept as that of Dijkstra’s algorithm but can handle negative edges as well. It has a better running time than that of Dijkstra’s algorithm.

e. PRIM'S ALGORITHM

Prim's algorithm finds a minimum spanning tree for a connected weighted graph. It implies that it finds a subset of edges that form a tree where the total weight of all the edges in the tree is minimized. it is sometimes called the DJP algorithm or Jerkin algorithm.

f. THE BUG ALGORITHM

The Bug algorithms, which are well-known navigation methods, are relatively efficient as they solve the navigation problem by saving only some points of the path curve and do not build full environment maps. As such, they are identical to the local planning techniques because they only need local environmental information but the robot needs to learn little of the global information. If the robot discovers that no such path exists, that is, a local minimum, the algorithms will terminate its motion and report that the target is unreachable.

2.2.1.1.3 MODEL PREDICTIVE TRAJECTORY GENERATOR

This algorithm is used for state lattice planner.

a. **PATH OPTIMIZATION SAMPLE**

Sample-path optimization is a method for optimizing limit functions occurring in stochastic modelling problems, such as steady-state functions in discrete-event dynamic systems.

b. **LOOKUP TABLE GENERATION SAMPLE**

This algorithm is presented for wheeled mobile robot trajectory generation that achieves a high degree of generality and efficiency. The generality derives from numerical linearization and inversion of forward models of propulsion, suspension, and motion for any type of vehicle. Efficiency is achieved by using fast numerical optimization techniques and effective initial guesses for the vehicle controls parameters. This approach can accommodate such effects as rough terrain, vehicle dynamics, models of wheel-terrain interaction, and other effects of interest.

FOLLOWING ARE THE MORE PATH PLANNING ALGORITHMS WHICH ARE PROFOUNDLY USED:

- STATE LATTICE PLANNING
- PROBABILISTIC ROAD-MAP PLANNING
- RAPIDLY EXPLORING RANDOM TREES
- CUBIC-SPINE PLANNING
- B-SPINE PLANNING
- ETA^3 SPLINE PATH PLANNING
- BEZIER PATH PLANNING and etc.

2.2.1.2 ALGORITHM USED

The Algorithm Used is Dijkstra's algorithm for finding the shortest path between nodes in a graph. The algorithm exists in many variants, Dijkstra's original algorithm found the shortest path between two given nodes, but a more common variant fixes a single node as the "source" node and finds shortest paths from the source to all other nodes in the graph, producing a shortest-path tree. Traffic information systems of these days use Dijkstra’s algorithm in order to track the source and destinations from a given particular source and destination. Dijkstra’s algorithm is used in SPF, shortest path first. There are two types of routing, Link State routing and Distance Vector routing. Dijkstra’s is based on Link State routing where each router calculates the shortest path to each node into the route table. Dijkstra’s Algorithm is a graph search algorithm that solves the single-source shortest path problem for a graph with non-negative edge path costs, producing a shortest path tree. The shortest distance between two points is a straight line. Dijkstra Algorithm is beneficial to smoothly navigate the robot in 2D Grid environments to avoid the obstacle without stopping in front of obstacle.

2.2.1.3 REASON TO CHOOSE THIS ALGORITHM

In this project we are using Dijkstra’s algorithm due to its efficiency. At first a few details. Dijkstra’s algorithm works by solving the sub problem k, which computes the shortest path from the source to vertices among the k closest vertices to the source. For the Dijkstra’s algorithm to work it should be directed- weighted graph and the edges should be non-negative. If the edges are negative then the actual shortest path cannot be obtained. At the k th round, there will be a set called Frontier of k vertices that will consist of the vertices closest to the source and the vertices that lie outside frontier are computed and put into New Frontier. The shortest distance obtained is maintained in sDist[w]. It holds the estimate of the distance from s to w. Dijkstra’s algorithm finds the next closest vertex by maintaining the New Frontier vertices in a priority-min queue. The algorithm works by keeping the shortest distance of vertex v from the source in an array, sDist. The shortest distance of the source to itself is zero. sDist for all other vertices is set to infinity to indicate that those vertices are not yet processed. After the algorithm finishes the processing of the vertices sDist will have the shortest distance of vertex w to s. two sets are maintained Frontier and New Frontier which helps in the processing of the algorithm. Frontier has k vertices which are closest to the source, will have already computed shortest distances to these vertices, for paths restricted up to k vertices. The vertices that resides outside of Frontier is put in a set called New Frontier.

The complexity/efficiency can be expressed in terms of Big-O Notation. Big-O gives another way of talking about the way input affects the algorithm’s running time. It gives an upper bound of the running time.

In Dijkstra’s algorithm, the efficiency varies depending on $|V|=n$ Delete Mins and $|E|$ updates for priority queues that were used.

If a Fibonacci heap was used then the complexity is $O(|E| + |V| \log |V|)$, which is the best bound. The Delete Mins operation takes $O(\log |V|)$.

If a standard binary heap is used then the complexity is O (| E |log |E|), | E | log |E| term comes from |E| updates for the standard heap.

If the set used is a priority queue then the complexity is O (|E|+|V| 2). O (V| 2) term comes from |V| scans of the unordered set New Frontier to find the vertex with the least sDist value.

2.2.1.4 ROBOT LOCALIZATION

Robot localization is the robot's ability to estimate its location relative to specific aspects within its environment, using whatever sensors are available. This process can be either relative localization or absolute localization.

a. Relative Localization

In relative localization, the robot calculates its current position relative to the previous locations, trajectories and velocities over a given period. As such, the robot requires knowledge of its initial location before it can continue determining its current location based on the direction, speed and time of its navigation. This method is implemented by using wheel encoders that count the revolutions of each wheel and an orientation sensor, such as electromagnetic compass that calculates the robot's direction. Because the robot measures its distance based on the start location, any error in the measurements resulting from the drift or slippage of the wheels will compound over time.

b. Absolute Localization

In the absolute localization method, the robot estimates its current position by determining the distance from predefined locations without regard to the previous location estimates. This has the advantage that the robot does not require prior information about the environment. However, the active landmarks' signals might be disturbed before being received by the robot and this will cause errors in the measurement. The Global Positioning System (GPS) is frequently used to measure the absolute position of robot that use active landmarks.

2.2.1.5 ROBOT LOCALIZATION METHODS

Mobile robots rely on localization systems that consistently and accurately determine its position and orientation as it moves through ever-evolving surroundings. In this project, we present a real-time positioning system for an mobile robot. We use the absolute positions by Global Positioning System (GPS) and the Inertial Measurement Unit (IMU) to provide reliable direction estimations to known the orientation of the robot movement. The sensor data are then fused by the Kalman Filter algorithm to obtain more accurate robot positions.

The localization techniques which can be used with a specific robotic depend on the available sensors. Sensors may deduce information about the environment, but measurements cannot be made with absolute certainty because these devices carry only partial or even incorrect information. Many aspects of the world cannot be directly measured; the exact location of the robot, for example, cannot be known deterministically, only inferred via the data derived from sensors. Further, all sensors are subject to persistent noise, which creates unpredictable perturbations in the expected readings. This noise has the effect of limiting the information which can be extracted. Sensors are also affected by errors unique to the given sensor itself. Further, GPS is prone to the effects of urban canyons or other physical obstructions which block or shadow the reception of satellite signals, thereby obfuscating readings, or preventing the arrival of sensor data all together.

Information from the surroundings is often used to update the internal state of the robot. A state can be understood as the collection of variables which provide information about the robot itself and the environment in which it operates, represented as a vector x of real numbers. Typically, state variables include the pose of the robot. A robot pose consists of the location and orientation of a robot relative to a global frame, which generally includes the coordinates in the Cartesian x, y, and z axis, and angular orientation which consists of roll, pitch and yaw. A robot may also contain the information regarding its velocity, and the velocity of its joints. Such variables are called the dynamic state, given their propensity to change with time. Further, the location and features of the surrounding environment are often accounted for among the other state information. Robots frequently possess the ability to perceive their surroundings. Regarding pose information, a robot must account for its position in three-dimensional space if it is to be able to navigate through it.
An inertial measurement unit (IMU) is an electronic device that measures and reports a body's specific force, angular rate, and sometimes the orientation of the body, using a combination of accelerometers, gyroscopes, and sometimes magnetometers. In a navigation system, the data reported by the IMU is fed into a processor which calculates attitude, velocity and position. A typical implementation referred to as a Strap Down Inertial System integrates angular rate from the gyroscope to calculate angular position. This is fused with the gravity vector measured by the accelerometers in a Kalman Filter to estimate attitude. The attitude estimate is used to transform acceleration measurements into an inertial reference frame (hence the term inertial navigation) where they are integrated once to get linear velocity, and twice to get linear position.

The most common implementations of sensor fusion belong to a class of algorithms called Kalman filters. A subset of Gaussian filters, Kalman filters are recursive state estimators which represent beliefs as multivariate normal distributions. Kalman filtering is used for many applications including filtering noisy signals, generating non-observable states, and predicting future states. Filtering noisy signals is essential since many sensors have an output that is to noisy too be used directly, and Kalman filtering lets your account for the uncertainty in the signal/state. One important use of generating non-observable states is for estimating velocity. It is common to have position sensors (encoders) on different joints; however, simply differentiating the position to get velocity produces noisy results. To fix this Kalman filtering can be used to estimate the velocity. Another nice feature of the Kalman filter is that it can be used to predict future states. This is useful when you have large time delays in your sensor feedback as this can cause instability in a motor control system.

2.2.2 PREVIOUS WORKS

A similar number of research paper on this domain has already been published. So, we can have a look in the research work by

1. WaiWai Maw, WaiPhyoEi in the International Journal of Science, Engineering and Technology Research (IJSETR) (Volume 6, Issue 11, November 2017, ISSN: 2278 - 7798). Our work is similar to this paper where the focus was on finding the Shortest Path and Tracking System Based on Dijkstra's Algorithm in Mobile Robot. The shortest path computations are one of the problems in graph theory. In shortest path problems, a directed weighted graph is given & the goal is to determine the shortest path among vertices. Dijkstra’s Algorithm is a graph search algorithm that solves the single-source shortest path problem for a graph with non-negative edge path costs, producing a shortest path tree. The shortest distance between two points is a straight line. But, in the real world, if those two points are located at opposite ends of the country, or even in different neighborhoods, it is unlikely to find a route that enables to travel from origin to destination via one straight road. Find out a map to determine the fastest way to drive somewhere, but these days, it is just as likely to use a Web-based service or a handheld device to help with driving directions. The popularity of mapping applications for mainstream consumer use once again has brought new challenges to the research problem known as the “shortest-path problem”, which is one of the fundamental quandaries in computing and graph theory, is intuitive to understand and simple to describe. In mapping terms, it is the problem of finding the quickest way to get from one location to another. Expressed more formally, in a graph in which vertices are joined by edges and in which each edge has a value, or cost, it is the problem of finding the lowest-cost path between two vertices, considered as a graph with positive weights. The nodes represent road junctions and each edge of considered as a graph with positive weights. The nodes represent road junctions and each edge of the graph is associated with a road segment between two junctions. The weight of an edge may correspond to the length of the associated road segment, the time needed to traverse the segment or the cost of traversing the segment. The simulation results for the proposed control scheme were carried out in that paper to find and track for the shortest path to demonstrate the performance of the whole system.
2. The work by Stephen Armah, Sun Yi and Taher Abu-Lebdeh (North Carolina A and T State University, Greensboro, NC 27411, USA) in the American Journal of Engineering and Applied Sciences 7 (1): 149-164, 2014 ISSN: 1941-7020. This study presents an effective navigation architecture that combines ‘go-to-goal’, ‘avoid-obstacle’ and ‘follow-wall’ controllers into a full navigation system. A MATLAB robot simulator is used to implement this navigation control algorithm. The robot in the simulator moves to a goal in the presence of convex and non-convex obstacles. Experiments are carried out using a ground mobile robot, Dr Robot X80SV, in a typical office environment to verify successful implementation of the navigation architecture algorithm programmed in MATLAB. The research paper also demonstrates algorithms to achieve tasks such as ‘move to a point’, ‘move to a pose’, ‘follow a line’, ‘move in a circle’ and ‘avoid obstacles’. A MATLAB robot simulator is used to implement the navigation control algorithm and the individual control algorithms were simulated using Simulink models. For the navigation control algorithm, the robot simulator is able to move to a goal in the presence of convex and non-convex obstacles. Also, several experiments are performed using a ground robot, Dr Robot X80SV, in a typical office environment to verify successful implementation of the navigation architecture algorithm programmed in MATLAB.

3. The work by Boru Diriba Hirpo and Prof. Wang Zhongmin (School of Mechanical Engineering, Tianjin University of Technology and Education) in International Journal of Engineering Research & Technology (IJERT) , (ISSN: 2278-0181) Vol. 6 Issue 10, October – 2017. This paper deals with Design and Control for Differential Drive Mobile Robot. The total mechanical structure of the wheeled robot platform was designed in detail and assembly drawing and the 3-D model was prepared. Also, the control of the robot platform is also mentioned. Kinematic based PID control system is used. The gains of PID controller is tuned to achieve the desired speed. SIMULINK model is prepared for DC motor and for whole robot system and the simulation result was discussed. MATLAB/SIMULINK package is used to model and simulate. This above paper gives us a very basic and detail idea of the Differential Drive which is a crucial part of our project.

4. Another work by Wael R. Abdul Majeed, (a Ph. D scholar in the Mechatronics Engineering Department / Al–Khwarizmi Engineering College / University of Baghdad) in International Journal of Computer Applications (0975 – 8887) Volume 91– No.2, April 2014. His work solves the problems of Simultaneous localization and mapping (SLAM) that deals with local path planning of an autonomous mobile robot in indoor environment, by using sonar sensors for object detection and range information, and also uses wheel encoders for tracking robot position and orientation based on dead - reckoning process. Although this robot work is of much advance level but we definitely learned a lot from this paper.

2.3 CONCLUSION

Researchers have developed many techniques to analyze images and detect objects but there are also limitations in these techniques. Subsequently, researchers have combined some of these techniques to achieve better results. In terms of the exploration path, the navigation is planned either globally or locally based on the algorithm used. Most algorithms assume that the robot has sufficient knowledge about the start and goal locations; its task is to find the optimal path to connect these two locations. Most researchers who have worked with mobile search robots assume that the searched area is completely known. In this project work we have assumed the environment to be known environment with the knowledge of the location of all the obstacles. The robot starts from the start location following the walls or obstacles to reach the destined location. For the unknown environment we have not taken the case of obstacles in the path, we only deal with start to final location.

3. PROPOSED SOLUTION

3.1 INTRODUCTION

In this project we are going to use wheeled robot. Wheeled robots are robots that navigate around the ground using motorized wheels to propel themselves. This design is simpler than using treads or legs and by using wheels they are easier to design, build, and program for movement in flat, not-so-rugged terrain. They are also more well controlled than other types of robots. Disadvantages of wheeled robots are that they can not navigate well over obstacles, such as rocky terrain, sharp declines, or areas with low friction. Wheeled robots are most popular among the consumer market, their differential steering provides low cost and simplicity. Robots can have any number of wheels, but three wheels are sufficient for static and dynamic balance. Additional wheels can add to balance; however, additional mechanisms will be required to keep all the wheels in the ground, when the terrain is not flat. In our project we have make four wheels robot for better stability.

3.2 MECHANICAL DESIGN

3.2.1 CHASIS

The robot's chassis was constructed from aluminium with a thickness of 3 mm. It is 300 mm long and 200 mm wide with a frame that includes two floors of rectangular shape. The bottom floor of the frame is used to attach the DC motors, microcontroller, motor drives, sensors that are placed at the front, and the battery. The upper floor is used to attach the raspberry pi and hold other range sensors that are placed at the robot's sides.

3.2.2 WHEEL CONFIGURATION

The decision was made to use a four-wheeled drive principle with a zero-turn radius mechanism. A robot using this configuration reorients itself by rotating the pair of wheels mounted on one side in a particular direction, while rotating the pair on the other side in the opposite direction. If the robot is required to drive in a straight line, it will rotate all the wheels at the same speed and in the same direction. This mechanism has some benefits with respect to other wheel configurations. The main benefits are:

- Robot stability: the four-wheeled configuration ensures stability;
- Robot movement: the robot can move to a desired site and turn in place to attain a particular orientation;
- Robot power: four wheels contribute to produce the robot's motion and steering; this makes the robot more powerful;
- Robot design: this kind of robot is easy to design and implement; and
- Robot steering radius: the robot is capable of executing a zero-point turn radius.

Although using four wheels with four motors has many advantages, it also has some disadvantages, the main ones being:

- Robot controllability: the straight-line motion control has proved to be difficult, since all the motors have to rotate at the same speed;

- Robot precision: the alignment of the four wheels must be precisely set to guarantee straight line motion. Thus, all wheels have to contact the ground at the same time to ensure straight line motion; and
- Robot drive: this method of robot motion needs strong motors on both sides to perform the zero-point turn radius.

3.2.3 MOTOR DRIVES

The previous section reports that four motors were first needed on the mechanical platform. The next step was choosing the motors. Three types of motor can be considered for driving the mobile robot: DC motors, stepper motors and servo motors. The decision was taken to use DC motors for the driving system. Initially, some specifications of DC motors were examined and determined. These motors must be able to drive the robot and produce motion.

To determine the power and torque needed by each motor, the maximum mass of the robot (*m*) was assumed to equal 10Kg and the robot will be used on a flat floor. Therefore, the force (*F*) needed to move the robot is

$F = F_f = \mu * m * g$

in which F_f and μ are the friction force and the estimated friction coefficient between the robot's wheels and the ground, respectively, while '*g*' is gravitational acceleration. Since the four-wheeled configuration is used, the robot's weight is applied on four wheels. In this case, the force supplied by each motor is

$F_{motor} = (\mu * m * g)/4$

If the maximum friction coefficient μ , when the robot starts moving, and *g* are assumed to equal 0.8 and 10 m/s^2 respectively, this will produce a force

$F_{motor} = (0.8*10*10)/4 = 20$ N

The maximum linear velocity of the robot is influenced by both the speed of the motors (measured in revolutions per minute (*RPM*)) and the diameter of the wheels (*d*). This is determined by the formula

$V_{max} = \Pi * d * RPM$

Since the mobile robot will be used to search the indoor environment, which might have maximum dimensions of 10 x 10m, then the robot's maximum speed (V_{max}) that is appropriate for the navigation and search tasks is assumed to be 10 m/minute. If the wheel diameter (*d*) is chosen to be 8 cm, then the motor's speed is

$10 = \Pi*0.08*RPM$

$RPM \approx 40$

In this case, the motor's maximum output power will be

$P_{max} = F_{motor} * V_{max}$

$P_{max} \approx 3.33$ Watt

The torque needed (*T*) is then calculated from

$T = P_{max}/(2\Pi*RPM*1/60)$

$T \approx 0.794$ Nm

The various calculations being completed, the choice of suitable motors can now be started. Four geared high-torque (1.17 Nm) DC motors that operate at 12 V were chosen. These motors have a maximum current draw of 1.5 Amps and rotate at 36 RPM. This is less than 40 RPM and reduces the robot's maximum speed to 9 m/minute, but it does not affect the calculations since the motor has a high torque that is greater than the maximum torque needed. The motors are attached to the robot's chassis and their shafts are directly coupled to wheels with 80 mm diameter. Each pair of motors, mounted on one side, turns in the same direction and at an identical speed. This is achieved by connecting each pair of wheels with the same line, as shown in figure

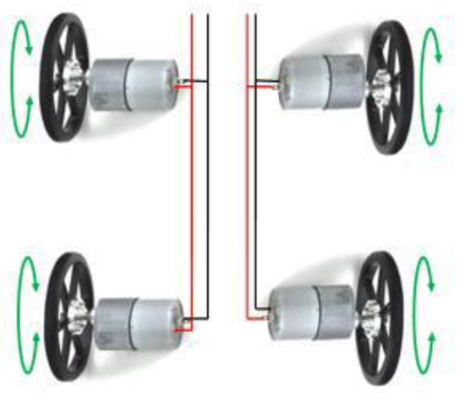


Fig.3.1.The wheeled-drive configuration

3.2.3.1 POWERING MOTORS

The speed and direction of the DC motors are controlled by the microcontroller board; however, the motors need more energy than can be supported by this board. Consequently, a separate motor drive amplifier that can support the required power and can also be controlled by the microcontroller must be used. The motors' control board, having 4 channels, each of which can control and provide 4 Amps (peak load) per motor, is chosen. This board provides pulse width modulation (PWM) pins that can individually control the speed and direction of four motors. Note that the PWM signals are generated using the microcontroller software and then sent to the pins of the motors' control board. As mentioned above, one of the requirements is that the robot is capable of execu ting a zero-turn radius. Accordingly, each pair of motors is connected to one channel and controlled together at the same time. For instance, if the robot wants to move in a straight line, it will run both pairs of wheels in the same direction and at identical speeds. Conversely, if it wants to change orientation, it will rotate both pairs of motors in opposite directions. The benefit of using this method is that the robot can change its direction by spinning about its central axis.

3.3 ELECTRONIC CIRCUIT DESIGN

Before describing the electronic system, it is worth mentioning the main requirement for designing a new robot. An autonomous mobile robot is needed to search for, find and relocate a target (object) in an indoor environment. In this scenario, it needs to employ sensors for navigating and detecting the target, and for controlling the robot's motion. An electronic module system will play the main role in the performance of these tasks. Typically, detecting the target requires use of a ultrasonic sensor after which it is controlled using control algorithm.

3.3.1 SOFTWARE

The software programs play the main role in implementing the robot's electronic system. The required software includes a choice of an operating system, the programming languages, and whatever tools and libraries are necessary.

PROGRAM LANGUAGE

The PYCHARM open-source software is used to edit, compile and download the program, which controls the robot's motion based on the sensory information and also control, the microcontroller.

3.3.2 MICROCONTROLLER

The Raspberry Pi was chosen because: it provides all the needs for sensors and motors; it has the open source software for Windows and Linux; and it is easy to configure I/O for other devices. The board has a USB port that is used for both downloading the control program for running on the board, and powering the board. The standard specification of areas, it is a credit card size single board computer with 1 GB ram and quad core processor, which deliver fast processing speed as compared to another microcontroller. Raspberry pi single board computer.

Fig.3.2.Microcontroller

3.4 POWER SYSTEM

The robot normally needs an DC power supply ±12 V. When this board is used for a robotics project, it must be run by a DC battery instead of the power supply. As such, a small DC-DC power supply, was used with a 12 V DC battery to provide the required voltages. The DC motors and servo motors require a 12 V and 6 V DC power supply. In this case, there were three options for using batteries as explained below:

- two DC batteries (12 V and 6 V) are used; however, this increases the robot's weight;
- a DC to DC converter is used; or
- DC battery cells are used to power the different parts.

3.5 CIRCUIT SCHEMATICS

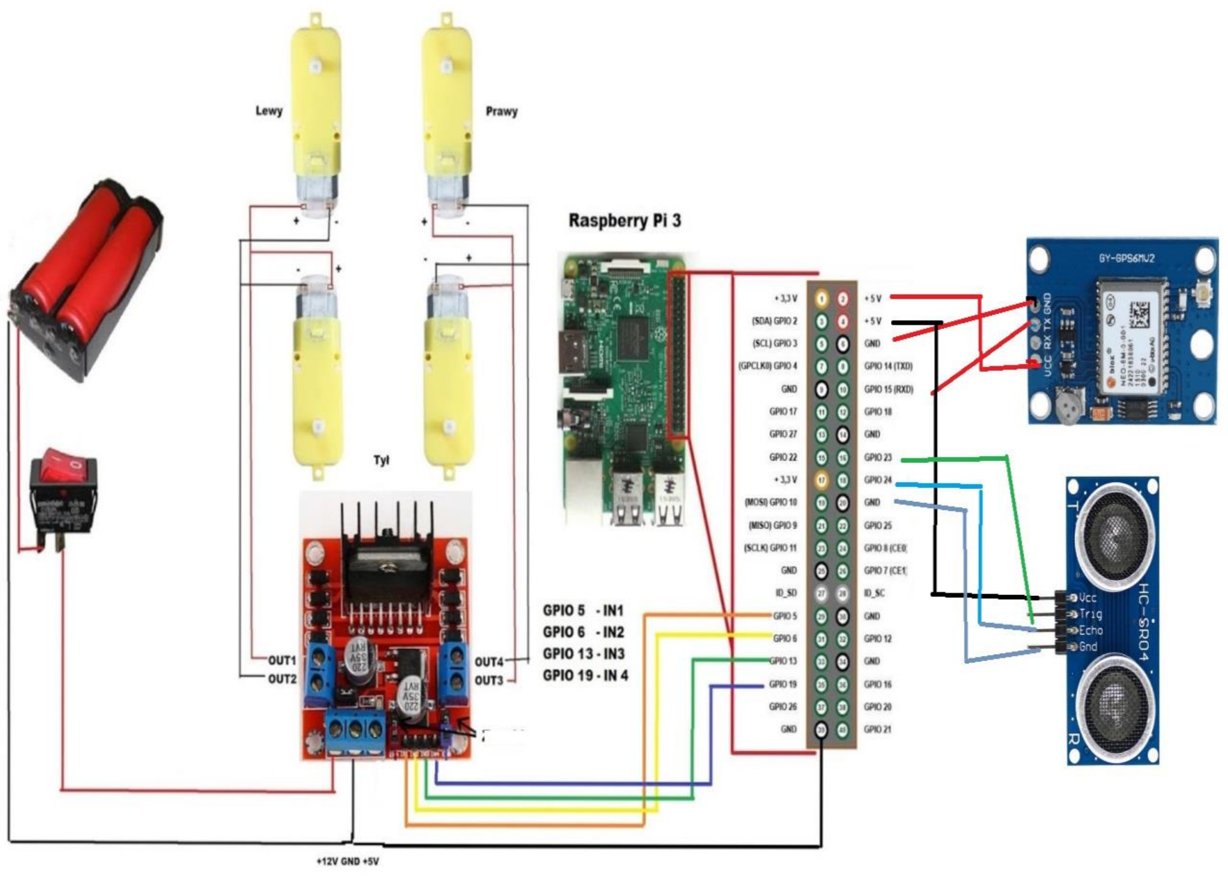


Fig.3.3.The circuit configuration

Figure shows all the components and data lines in the electronic circuitry of the robot.

.

- The microcontroller fed the logic circuit of the motor controller via two wires (Vcc and Ground).
- Each pair of motors, attached to one side of the robot, was connected to one channel of the motors’ control board
- The robot’s power was provided by 4 × 1.2-volt cells connected in series. The DC motor controller used all the cells (12 volts).

3.6 SUMMARY

This chapter proposes a methodology for designing and implementing a wheeled robot. The designed robot has a four-wheeled drive principle with a zero-turn radius mechanism. The robot’s electronic system has on module which is microcontroller which makes the decision to make the turn.

4 PRACTICAL IMPLEMENTATIONS

4.1 INTRODUCTION

Today's era demands a friendlier relation between man and machines. This project is an effort to build that relationship. Our self-navigating rover is mainly designed to go to areas where human intervention can be dangerous or disastrous. Manpower is the one of the major possibilities f or the development of our nation. And our project is focussed on saving this possibility only. Robots can go to the desired location and search for the proposed target and this way it contributes in saving man from entering a life losing position. This project focus on developing a mechanism that would not only save life but also can make the development much faster and easier.

4.2 FEEDBACK CONTROL ANALYSIS

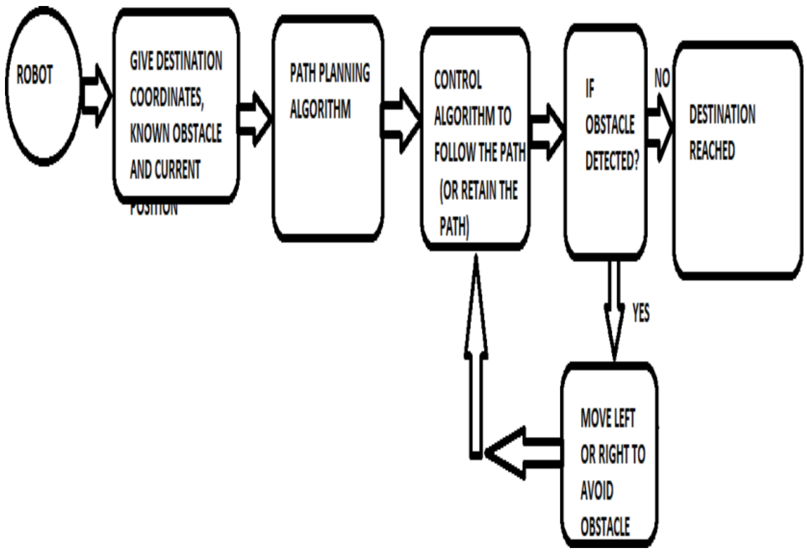

ROBOT
GIVE DESTINATION COORDINATES, KNOWN OBSTACLE AND CURRENT POSITION
PATH PLANNING ALGORITHM
CONTROL ALGORITHM TO FOLLOW THE PATH (OR RETAIN THE PATH)
IF OBSTACLE DETECTED?
NO
DESTINATION REACHED
YES
MOVE LEFT OR RIGHT TO AVOID OBSTACLE

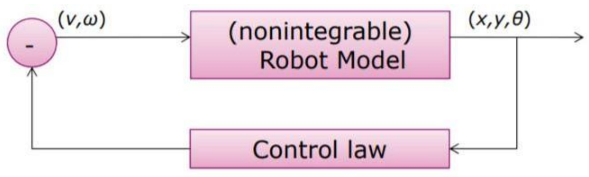
Fig.4.1.Feedback control analysis

The proposed work suggests that at first the rover will be given the destination coordinates. At first, the known obstacles will be given to the path planning algorithm. The path planning algorithm will use these obstacles, the destination coordinates and the current position to give the path to reach destination coordinates. As soon as the path is known the rover will try to follow it using control

$$\begin{bmatrix} v(t) \\ \omega(t) \end{bmatrix} = K \cdot e = K \cdot {}^{R}\begin{bmatrix} x \\ y \\ \theta \end{bmatrix}$$

algorithm. The input to the control algorithm will be x, y, and Θ which the robot will get from the accelerometer, gyro sensors. The output of the control algorithm will be v and ω.

The control of v(t) and ω(t) is such that it drives the error to zero



This v and ω will be distributed to the left and right wheels to generate torque as given by the below equations

$$v = \frac{r\dot{\phi}_r}{2} + \frac{r\dot{\phi}_l}{2}$$

$$\omega = \frac{r\dot{\phi}_r}{2l} - \frac{r\dot{\phi}_l}{2l}$$

Where r is the radius of the wheels, l is the length between the wheels and φ is angular displacement of the wheels.

4.3 KINEMATICS POSITION CONTROL

Now the kinematics position control will be used to follow the path provided. To understand it the below picture is provided.

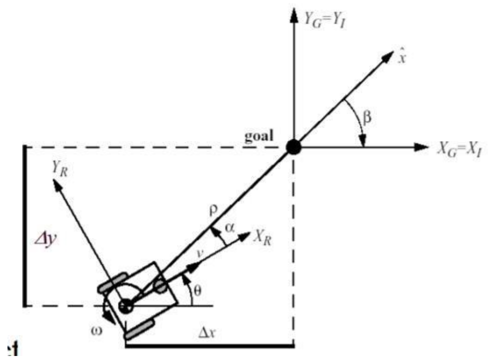


Fig.4.2 .kinematics position control

The kinematics of a differential drive mobile robot described in the inertial frame { xI, yI, q } is given by

$$^{I}\begin{bmatrix}\dot{x}\\ \dot{y}\\ \dot{\theta}\end{bmatrix} = \begin{bmatrix}cos\theta & 0\\ sin\theta & 0\\ 0 & 1\end{bmatrix}\begin{bmatrix}v\\ \omega\end{bmatrix}$$

Where x and y are the linear velocities in the direction of the xI and yI of the inertial frame. Let alpha denote the angle between the xR axis of the robots' reference frame and the vector connecting the centre of the axle of the wheels with the final position.

Coordinates transformation into polar coordinates with its origin at goal position:

$$\rho = \sqrt{\Delta x^2 + \Delta y^2}$$

$$\alpha = -\theta + \operatorname{atan2}(\Delta y, \Delta x)$$

$$\beta = -\theta - \alpha$$

System description, in the new polar coordinates

$$\begin{bmatrix}\dot\rho\\ \dot\alpha\\ \dot\beta\end{bmatrix} = \begin{bmatrix}-\cos\alpha & 0\\ \frac{\sin\alpha}{\rho} & -1\\ -\frac{\sin\alpha}{\rho} & 0\end{bmatrix}\begin{bmatrix}v\\ \omega\end{bmatrix} \quad \text{for } I_1 = \left(-\frac{\pi}{2}, \frac{\pi}{2}\right]$$

$$\begin{bmatrix}\dot\rho\\ \dot\alpha\\ \dot\beta\end{bmatrix} = \begin{bmatrix}\cos\alpha & 0\\ -\frac{\sin\alpha}{\rho} & 1\\ \frac{\sin\alpha}{\rho} & 0\end{bmatrix}\begin{bmatrix}v\\ \omega\end{bmatrix} \quad \text{for } I_2 = (-\pi, -\pi/2] \cup (\pi/2, \pi]$$

The coordinates transformation is not defined at x = y = 0; For α=I1 the forward direction of the robot points toward the goal, for α=I2 it is the backward direction.

$$\alpha \in I_1 \qquad I_1 = \left(-\frac{\pi}{2}, \frac{\pi}{2}\right]$$

By properly defining the forward direction of the robot at its initial configuration, it is always possible to have at t=0. However, this does not mean that a remains in I1 for all time t.

It can be shown, that with $v = k_\rho \rho$ $\omega = k_\alpha \alpha + k_\beta \beta$ the feedback-controlled system

$$\begin{bmatrix}\dot\rho\\ \dot\alpha\\ \dot\beta\end{bmatrix} = \begin{bmatrix}-k_\rho \rho \cos\alpha\\ k_\rho \sin\alpha - k_\alpha \alpha - k_\beta \beta\\ -k_\rho \sin\alpha\end{bmatrix}$$

will drive the robot to (ρ, α, β) = (0, 0, 0). This is how the robot will follow the path.

If an obstacle comes in the way of the robot then using the ultrasonic sensor it will detect the obstacle and will know the obstacle position. After that using the control algorithm it will avoid the obstacle and will try to retain its original path.

4.4 ROBOT MOVEMENTS

4.4.1 NAVIGATION STRATEGY

Finding location on map: - To find out initial position of rover on the Google map and to move from source to destination based on GPS data that is latitude and longitude are accessed through installed GPS sensor on Raspberry pi board. Once the initial position of rover is marked it will move towards destination in autopilot mode designed for self-navigation at successful collaboration of the GPS sensor with raspberry pi it provides GPS data in the form of NMEA standard. It is required to extract longitude and latitude information from the NMEA sentences. For example, suppose that the NMEA string values for a position are as follows.

1954.7704, N, 07521.1704, E

As it turns out, the NMEA strings are decimal-decimal values. In order to turn them into decimal degree values, we have to do following simple operation:

1954.7704 N=19 degrees+54.7704/60=19.91284N
07521.1704E=07 degrees+521.1704/60=75.35284 E.

4.4.2 OBSTACLE DETECTION

The ultrasonic sensor is mounted over the rover to check obstacles with the help of ultrasonic waves. This sensor continuously transmits ultrasonic waves ahead the rover if obstacle occur that waves reflected as the input signal to rover to avoid the obstacle coming in the path of rover. The ultrasonic sensor is consisting from multi vibrator fixed at its base. Resonator and vibrator are used to make the multi vibrator. Ultrasonic sensor consists of two parts, emitter and detector. Emitter produces 40 KHz sound wave and detector which detect
the sound wave and send signals to raspberry pi. Eqn. (1) is used to calculate the distance of obstacles from rover, which are used to detect the obstacle coming in the path of rover.

Distance = (time x speed of sound wave)/2 ….. (1)

4.4.3 COMPUTERIZED DECISION MAKING

Ultrasonic sensor work with transmission and receiver, transmitter transmit the ultrasonic wave in front of the robot and when obstacle occurred in the path of robot ultra-sonic wave reflected and received by the receiver which is input to raspberry pi to take a decision as per the received signals. According to the signal raspberry pi take the decision with respect to forward, backward, left and right to avoid the obstacle coming in the path of the rover. Proposed rover will capable to take self-decision to avoid the obstacle per ultrasonic signals.

4.5 ALGORITHM

4.7.1 MOTOR CONTROL ALGORITHM

1. Step 1: **Input** Turn power on to Dc Geared Motors
2. Step 2: **Output** Rotation of motors
3. Step 3: **Begin**
4. Step 4: Declared four GPIO pins to control motor such as GPIO pin 1, pin 2, pin 3 and pin 4 in which pin 1 and pin 2 are used to control the motor number 1 and pin 3 and pin 4 are used to control the motor number 2. Also, Declared Enable pin A and Enable pin B.
5. Step 5: **IF** (GPIO pin 1, pin 3=1) **AND** (Enable pin A, Enable pin B=True)
6. Step 6: Move Forward
7. Step 7: **IF** (GPIO pin 1, pin 4=1) **AND** (Enable pin A, Enable pin B=True)
8. Step 8: Move Backward
9. Step 9: **IF** (GPIO pin 1, pin 3=1) **AND** (Enable pin A=True **AND** Enable pin B=False)
10. Step 10: Take left turn
11. Step 11: **IF** (GPIO pin 1, pin 3=1) **AND** (Enable pin A=False **AND** Enable pin B=True)
12. Step 12: Take right turn
13. Step 13: **IF** (GPIO pins 1, 2, 3, 4=0) **AND** (Enable pin A=False **AND** Enable pin B=False)

Step 14: **End**

4.5.2 GPS LOCATION ALGORITHM

1. Step 1: **Input** Turn power on to GPS sensor
2. Step 2: **Output** show location on Google map
3. Step 3: **Begin**
4. Step 4: Data acquisition is done
5. Step 5: Extract the latitude and longitude form NMEA GPS data format
6. Step 6: Save latitude and longitude of GPS sensor
7. Step 7: Post latitude and longitude to server
8. Step 8: Show initial position of Rover on Google map
9. Step 9: **END.**

4.5.3 COMPUTERIZED DECISION MECHANISM ALGORITHM

1. Step 1: Input Turn Power on Raspberry pi
2. Step 2: Output Self-decision mechanism of obstacle avoidance
3. Step 3: Begin
4. Step 4: While True:
5. Distance=Get Distance ()
6. Step 5: IF (Distance== 6)
7. Stop
8. Step 6: ELIF (Distance <=15 AND Distance >=7)
9. Stop
10. Delay (1)
11. Move Backward
12. Delay (1)
13. Stop
14. Take turn right
15. Delay (1)
16. Stop
17. Step 7: Right Distance=Get right distance ()
18. Move Backward
19. Delay (1)
20. Take turn left
21. Step 8: Left Distance=Get left distance ()
22. Step 9: IF (Right Distance <=Left distance)
23. Take left turn
24. Delay (1)
25. Move Forward
26. Delay (1)
27. Step 10: ELIF (Right Distance >=Left distance)
28. Take turn right
29. Delay (1)
30. Move Forward
31. Delay (1)
32. Step 11: **END**.

4.5.4 DIJKSTRA'S ALGORITHM

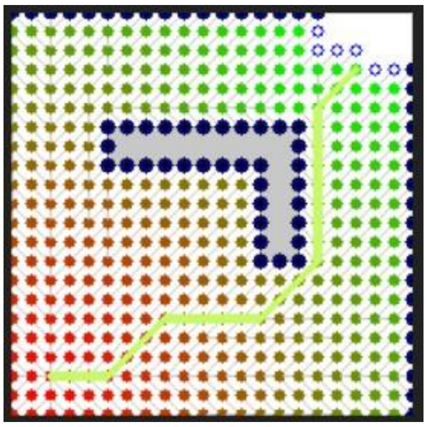


Fig.4.3.Dijkstra’s Algorithm

Let the node at which we are starting be called the initial node. Let the distance of node *Y* be the distance from the initial node to *Y*. Dijkstra's algorithm will assign some initial distance values and will try to improve them step by step.

1. Mark all nodes unvisited. Create a set of all the unvisited nodes called the *unvisited set*.
2. Assign to every node a tentative distance value: set it to zero for our initial node and to infinity for all other nodes. Set the initial node as current.
3. For the current node, consider all of its unvisited neighbors and calculate their *tentative* distances through the current node. Compare the newly calculated *tentative* distance to the current assigned value and assign the smaller one. For example, if the current node *A* is marked with a distance of 6, and the edge connecting it with a neighbor *B* has length 2, then the distance to *B* through *A* will be 6 + 2 = 8. If B was previously marked with a distance greater than 8 then change it to 8. Otherwise, the current value will be kept.
4. When we are done considering all of the unvisited neighbors of the current node, mark the current node as visited and remove it from the *unvisited set*. A visited node will never be checked again.
5. If the destination node has been marked visited (when planning a route between two specific nodes) or if the smallest tentative distance among the nodes in the *unvisited set* is infinity (when planning a complete traversal; occurs when there is no connection between the initial node and remaining unvisited nodes), then stop. The algorithm has finished.
6. Otherwise, select the unvisited node that is marked with the smallest tentative distance, set it
as the new "current node", and go back to step 3.

When planning a route, it is actually not necessary to wait until the destination node is "visited" as above: the algorithm can stop once the destination node has the smallest tentative distance among all "unvisited" nodes (and thus could be selected as the next "current").

4.6 EXPERIMENTAL WORK

Proposed rover is implemented using raspberry pi framework. Raspberry pi having its own operating system which is Raspbian and it will give the fast processing speed as compared to micro-controller. In raspberry pi, there is no limit of memory as like micro-controller and we run large programs using raspberry pi. Raspberry pi having 40 pins for attachment of motor driver kit and various sensors.

4.7 PYTHON CODE: -

```
import math
import matplotlib.pyplot as plt

show_animation = True




class Dijkstra:


    def __init__(self, ox, oy, resolution, robot_radius):
        """
        Initialize map for a star planning

        ox: x position list of Obstacles [m]
        oy: y position list of Obstacles [m]
        resolution: grid resolution [m]
        rr: robot radius[m]
        """


        self.min_x = None
        self.min_y = None
        self.max_x = None
        self.max_y = None
        self.x_width = None
        self.y_width = None
        self.obstacle_map = None


        self.resolution = resolution
        self.robot_radius = robot_radius
        self.calc_obstacle_map(ox, oy)
        self.motion = self.get_motion_model()


    class Node:
        def __init__(self, x, y, cost, parent):
            self.x = x  # index of grid
            self.y = y  # index of grid
            self.cost = cost
            self.parent = parent  # index of previous Node


        def __str__(self):
            return str(self.x) + "," + str(self.y) + "," + str(
                self.cost) + "," + str(self.parent)


    def planning(self, sx, sy, gx, gy):
        """
        dijkstra path search

        input:
            sx: start x position [m]
            sy: start y position [m]
            gx: goal x position [m]
            gx: goal x position [m]

        output:
            rx: x position list of the final path
            ry: y position list of the final path
        """


        start_node = self.Node(self.calc_xy_index(sx, self.min_x),
                               self.calc_xy_index(sy, self.min_y), 0.0, -1)
        goal_node = self.Node(self.calc_xy_index(gx, self.min_x),
                              self.calc_xy_index(gy, self.min_y), 0.0, -1)


        open_set, closed_set = dict(), dict()
        open_set[self.calc_index(start_node)] = start_node
```

```
while 1:
    c_id = min(open_set, key=lambda o: open_set[o].cost)
    current = open_set[c_id]


    # show graph
    if show_animation:  # pragma: no cover
        plt.plot(self.calc_position(current.x, self.min_x),
                 self.calc_position(current.y, self.min_y), "xc")
        # for stopping simulation with the esc key.
        plt.gcf().canvas.mpl_connect(
            'key_release_event',
            lambda event: [exit(0) if event.key == 'escape' else None])
        if len(closed_set.keys()) % 10 == 0:
            plt.pause(0.001)


    if current.x == goal_node.x and current.y == goal_node.y:
        print("Find goal")
        goal_node.parent = current.parent
        goal_node.cost = current.cost
        break


    # Remove the item from the open set
    del open_set[c_id]


    # Add it to the closed set
    closed_set[c_id] = current


    # expand search grid based on motion model
    for move_x, move_y, move_cost in self.motion:
        node = self.Node(current.x + move_x,
                         current.y + move_y,
                         current.cost + move_cost, c_id)
        n_id = self.calc_index(node)


        if n_id in closed_set:
            continue
            if not self.verify_node(node):
                continue


            if n_id not in open_set:
                open_set[n_id] = node  # Discover a new node
            else:
                if open_set[n_id].cost >= node.cost:
                    # This path is the best until now. record it!
                    open_set[n_id] = node


    rx, ry = self.calc_final_path(goal_node, closed_set)


    return rx, ry


def calc_final_path(self, goal_node, closed_set):
    # generate final course
    rx, ry = [self.calc_position(goal_node.x, self.min_x)], [
        self.calc_position(goal_node.y, self.min_y)]
    parent = goal_node.parent
    while parent != -1:
        n = closed_set[parent]
        rx.append(self.calc_position(n.x, self.min_x))
        ry.append(self.calc_position(n.y, self.min_y))
        parent = n.parent


    return rx, ry


def calc_position(self, index, minp):
    pos = index * self.resolution + minp
    return pos


def calc_xy_index(self, position, minp):
    return round((position - minp) / self.resolution)
```

```
def calc_index(self, node):
    return (node.y - self.min_y) * self.x_width + (node.x - self.min_x)


def verify_node(self, node):
    px = self.calc_position(node.x, self.min_x)
    py = self.calc_position(node.y, self.min_y)


    if px < self.min_x:
        return False
    if py < self.min_y:
        return False
    if px >= self.max_x:
        return False
    if py >= self.max_y:
        return False


    if self.obstacle_map[node.x][node.y]:
        return False


    return True


def calc_obstacle_map(self, ox, oy):


    self.min_x = round(min(ox))
    self.min_y = round(min(oy))
    self.max_x = round(max(ox))
    self.max_y = round(max(oy))
    print("min_x:", self.min_x)
    print("min_y:", self.min_y)
    print("max_x:", self.max_x)
    print("max_y:", self.max_y)


    self.x_width = round((self.max_x - self.min_x) / self.resolution)
    self.y_width = round((self.max_y - self.min_y) / self.resolution)
    print("x_width:", self.x_width)
    print("y_width:", self.y_width)
```

```
        # obstacle map generation
        self.obstacle_map = [[False for _ in range(self.y_width)]
                             for _ in range(self.x_width)]
        for ix in range(self.x_width):
            x = self.calc_position(ix, self.min_x)
            for iy in range(self.y_width):
                y = self.calc_position(iy, self.min_y)
                for iox, ioy in zip(ox, oy):
                    d = math.hypot(iox - x, ioy - y)
                    if d <= self.robot_radius:
                        self.obstacle_map[ix][iy] = True
                        break


    @staticmethod
    def get_motion_model():
        # dx, dy, cost
        motion = [[1, 0, 1],
                  [0, 1, 1],
                  [-1, 0, 1],
                  [0, -1, 1],
                  [-1, -1, math.sqrt(2)],
                  [-1, 1, math.sqrt(2)],
                  [1, -1, math.sqrt(2)],
                  [1, 1, math.sqrt(2)]]


        return motion




def main():
    print(__file__ + " start!!")


    # start and goal position
    sx = -5.0  # [m]
    sy = -5.0  # [m]
    gx = 50.0  # [m]
    gy = 50.0  # [m]
```

```
grid_size = 2.0  # [m]
robot_radius = 1.0  # [m]


# set obstacle positions
ox, oy = [], []
for i in range(-10, 60):
    ox.append(i)
    oy.append(-10.0)
for i in range(-10, 60):
    ox.append(60.0)
    oy.append(i)
for i in range(-10, 61):
    ox.append(i)
    oy.append(60.0)
for i in range(-10, 61):
    ox.append(-10.0)
    oy.append(i)
for i in range(-10, 40):
    ox.append(20.0)
    oy.append(i)
for i in range(0, 40):
    ox.append(40.0)
    oy.append(60.0 - i)


if show_animation:  # pragma: no cover
    plt.plot(ox, oy, ".k")
    plt.plot(sx, sy, "og")
    plt.plot(gx, gy, "xb")
    plt.grid(True)
    plt.axis("equal")


dijkstra = Dijkstra(ox, oy, grid_size, robot_radius)
rx, ry = dijkstra.planning(sx, sy, gx, gy)


if show_animation:  # pragma: no cover
    plt.plot(rx, ry, "-r")
    plt.pause(0.01)
    plt.show()
```

```
if __name__ == '__main__':
    main()
```

5. RESULTS

5.1 SIMULATION RESULT

The main control program and algorithms were written using python. Figure (a) represents the environmental map of the room. The green curve is the path that will be followed by the robot.

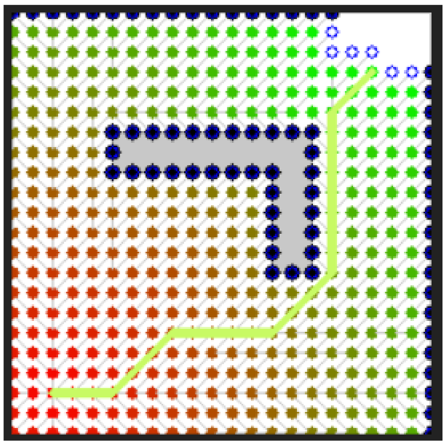


Fig:5.1 Dijkstra’s algorithm

5.2 SUMMARISED RESULT

The optimal navigation system must support three skills for the robot. The first is path-planning skill, which is the robot’s ability to find the shortest way from starting point to the target position, while avoiding obstacles. All the existing navigation algorithms assume that the robot knows the exact location of both places.

Another aspect of the project is how should the different types of sensors be integrated within the control system for the robot?
The accuracy of a measurement system will dramatically rise if the robot is equipped with the high number of sensors. However, this increases the robot’s price and leads to a more complex control system in its implementation. Therefore, the number of sensors must be reduced without affecting the effectiveness of the robot motion.

Various sensors can be used to enable the robot to sense its surrounding environment and then decide on its behaviour. These sensors provide the environmental information by means of electrical signals, which must be processed in the robot’s processor to generate meaningful information for influencing the robot’s motion. Some types of sensors need processors with a high computation capacity to analyse their signals. Typically, the robot is equipped with the appropriate types of sensors based on its application, and the robot’s environmental situation. In this project, the robot will follow a path avoiding obstacls so we add sensors to detect obstacles.

Generally, the robot navigates within its environment, which contains a starting location, target location and number of arbitrarily sized and shaped obstacles. Its objective is to move from the start to the target, without any collision., Dijkstra’s algorithm is used to find the shortest path for directed and undirected graph. Simulation results using Dijkstra's algorithm for any kind of graphs are successfully implemented in PYCHARM. In conclusion, the proposed method has been implemented in a shortest path tracking system so that the implement controller has the characteristics of high modularity and probability. By satisfying these facts, the robot can reach to its final point without any error in distance and direction. In addition, these mobile robots can be used for communications, transportation, and electronics problem by moving from source point to target point in desired distance and direction in order to find the shortest path.

The user can see the generalized coordinates vector in the inertial frame from the output results of kinematic model for mobile robot. The simulation result is the shortest path tracking of movement. The movement along X axis and Y axis. The robot moves from start node to desired node along the shortest path. For this movement, we can see the error position and the robot can reach or not to its desired node exactly.

.

5.3 PROBLEMS AND LIMITATIONS

It can be only used in known environment. As it follows a predetermined path so any physical obstacle randomly comes in path then it will not be able to move. Size of the chassis is not big enough to contain all the parts easily and in future addi8ng extra equipment’s will be a tough job.

6. CONCLUSION AND FUTURE WORK

6.1 CONCLUSIONS

The proposed rover is capable to take a self-decision and move in an unfamiliar environment without any human interaction.

Autonomous real time obstacle avoidance is achieved from proposed rover, it also detects obstacles accurately and take decision to move to
left, right, forward and backward as per the availability of sufficient space without colliding with surrounding.

Ultrasonic sensor mounted over the rover transmits 40 KHz Frequency towards object and if object is laying in the path, it reflects the sound wave and obstacles is detected.

Designed rover gives 100% obstacle detection and avoidance accuracy in well light and dimly lightning condition with single
solid obstacle.

The mobile robot is supposed to navigate on the roads in real time and reach the predetermined goal while discovering and detecting.

A complete modeling and path determination of the intersection environments for autonomous mobile robot navigation is derived and presented.

The proposed research can be considered in future to improve the implementation of self-navigation rover with various sensors such as Gas detection sensor, Temperature sensor, humidity sensor etc. Also collaborate with high quality GPS and compass sensor which avoid the distortion of GPS signals in case of low altitude.

6.2 FUTURE WORK

Today's era demands a friendlier relation between man and machines. This project is an effort to build that relationship. Our self-navigating rover is mainly designed to go to areas where human intervention can be dangerous or disastrous. Manpower is the one of the major possibilities for the development of our nation. And our project is focussed on saving this possibility only. Robots can go to the desired location and search for the proposed target and this way it contributes in saving man from entering a life losing position.

This project focusses on developing a mechanism that would not only save life but also can make the development much faster and easier. The major areas where this project can be a boon to the society are explained below: -

- In high temperature regions, such as coal mines, use of manpower for basic work is dangerous as well as inefficient. Self-navigating rovers can go to the desired target to fulfil the proposed task.

- During surface operations on Mars, self-navigating rover receives a new set of instructions at the beginning of each sol. Sent from the scientists and engineers on Earth, the command sequence tells the rover what targets to go to and what science experiments to perform on Mars. Development of our project can help scientists as this rover can sense the obstacles without getting damaged and hence through satellites can report it as well.

- This project can be a useful household tool as well. In a time where the whole world is struggling against COVID 19 our developed robot can serve the people in providing their necessities at the door with a much lesser risk of people getting infected.

- Regions where midnight guarding is required it becomes difficult for humans as their visibility becomes an issue there. Our project can be a source to maintain the safety from ground level that is any particularly small isolated location to top level such as defence purpose if developed in the correct direction.

- Places where human sustainability is doubtful but some research needs to be performed, we can use this self-navigating rover to perform the tasks and in this way this rover proves their use as a human alternative. As we know about the Chernobyl disaster in 1986, there human exposure to such highly radioactive graphite was not possible and thus robots were used for the safety measures robots were used to remove the graphite. Our developed project provides an advanced solution for such future problems.

Our proposed project can act as a friend to people who have lost their visibility in guiding directions.

7. REFERENCE

1. Kumar RC, Khan S, Kumar D, Birua R, Mondal S, et al. (2013) Obstacle Avoiding Robot-a Promising One. International Journal of Advanced Research in Electrical, Electronics and Instrumentation Engineering 2: 1430-1434.

2. Carullo A, Parvis M (2001) An Ultrasonic Sensor for Distance Measurement in Automotive Applications. IEEE Sensors Journal.

3. Bhagat K, Deshmukh S (2016) Obstacle Avoidance Robot. International Journal of Science, Engineering and Technology Research (IJSETR).

4. Ankit V, Jigar P, Savan V (2016) Obstacle Avoidance Robotic Vehicle Using Ultrasonic Sensor, Android and Bluetooth for Obstacle Detection. International Research Journal of Engineering and Technology (IRJET) 3: 339-348.

5. Pannu GS, Dawud M, Gupta P (2015) Design and Implementation of Autonomous Car using Raspberry Pi. International Journal of Computer Applications 113: 22-29.

6. Hanumante V, Roy S, Maity S (2013) Low Cost Obstacle Avoidance Robot. International Journal of Soft Computing and Engineering (IJSCE) 3: 52-55.

7. Aman MS, Mahmud MA, Jiang H, Abdelgawad A, Yelamarthi K (2016) A sensor fusion methodology for obstacle avoidance robot. Electro Information Technology (EIT).

8. Rahman A, Aslam MF, Ejaz H (2014) GPS based Navigation and Collision Avoidance System using Ultrasonic Sensors and Image Processing for Autonomous Vehicle. International Journal of Computer and Electronics Research.

9. Marques L, Rachkov M, Almeida AT (2002) Mobile pneumatic robot for demining. Proceedings of the 2002 IEEE International Conference on Robotics and Automation Washington.

10. Al-Faiz MZ, Mahameda GE (2015) GPS-based Navigated Autonomous Robot. International Journal of Emerging Trends in Engineering Research 3: 1-7.

11. Liu Y, Gao J, Shi X, Cao X, Zhao F, et al. (2016) Navigation research on outdoor miniature reconnaissance robot. IEEE International Conference on Mechatronics and Automation (ICMA).