LOCAL PATH PLANNING OF A MOBILE ROBOT USING LASER RANGE SENSOR BASED ON MODIFIED POINT BUG ALGORITHM

BY

BISHWADEB SINGHA
B. TECH (MEECHANICAL ENGINEERING),2019
HALDIA INSTITUTE OF TECHNOLOGY

EXAMINATION ROLL NO: M4PRD22007
REGISTRATION NO: 154475 of 2020-2021

THESIS

SUBMITTED IN PARTIAL FULFILMENT OF THE REQUIREMENTS FOR THE AWARD OF THE DEGREE OF MASTER OF PRODUCTION ENGINEERING IN THE FACULTY OF ENGINEERING AND TECHNOLOGY
JADAVPUR UNIVERSITY
2022

DEPARTMENT OF PRODUCTION ENGINEERING
JADAVPUR UNIVERSITY
KOLKATA-700032

JADAVPUR UNIVERSITY FACULTY OF ENGINEERING AND TECHNOLOGY

CERTIFICATE OF RECOMMENDATION

I HEREBY RECOMMEND THAT THE THESIS ENTITLED "**LOCAL PATH PLANNING OF A MOBILE ROBOT USING LASER RANGE SENSOR BASED ON MODIFIED POINT BUG ALGORITHM**" SUBMITTED BY **BISHWADEB SINGHA** CARRIED OUT UNDER MY SUPERVISION AND GUIDANCE BE ACCEPTED IN THE PARTIAL FULFILLMENT OF THE REQUIREMENTS FORTHE DEGREE OF "**MASTER OF ENGINEERING IN PRODUCTION ENGINEERING**" IN JADAVPUR UNIVERSITY.

(Mr. Subir Kr. Debnath)
Thesis Advisor
Dept. of Production Engineering
Jadavpur University
Kolkata-700032

HEAD, Dept. of Production Engineering
Jadavpur University
Kolkata-700032

DEAN, Faculty of Engineering and Technology
Jadavpur University
Kolkata-700

JADAVPUR UNIVERSITY

FACULTY OF ENGINEERING AND TECHNOLOGY

CERTIFICATE OF APPROVAL

The foregoing thesis is hereby approved as a creditable study of an engineering subject carried out and presented in a manner of satisfactory to warrant its acceptance as a prerequisite to the degree for which it has been submitted. It is understood that by this approval, the undersigned do not necessarily endorse or approve any statement made, opinion expressed and, conclusions drawn therein but approve the thesis only for the purpose for which it has been submitted.

COMMITE OF FINAL EXAMINATION FOR

EVALUATION OF THE PROJECT WORK ……………………………………

(External examiner)

………………………………………

(Internal examiner)

ACKNOWLEDGEMENT

It is my greatest fortune to perform the thesis work in the Production Engineering Department, Jadavpur University. I express my heartfelt gratitude to my respected teacher, mentor and thesis supervisor Mr. Subir Kr. Debnath, Associate Professor, Department of Production Engineering, Jadavpur University.

Special thanks to Prof. Ajoy Kumar Dutta, Professor, Department of Production Engineering, Jadavpur University, for his encouragement and kind support.

I also express my deep respect to all the faculties of the Production Engineering Department, Jadavpur University. I am greatly thankful to them for their constant motivation.

I am also thankful to the librarian and technicians of our department for their cordial assistance.

Finally, I am deeply indebted to my parents, for their moral support and continuous encouragement while carrying out this study.

Any omission in this brief acknowledgment does not mean a lack of gratitude.

…………………………..

(BISHWADEB SINGHA)

Class Roll No: 002011702008

TABLE OF CONTENTS

CHAPTER-1

1.0 INTRODUCTION

1.1 INTRODUCTION TO MOBILE ROBOT

Mobile robots can move autonomously (in an industrial plant, laboratory, planetary surface, etc.), that is, without assistance from external human operators. A robot is autonomous when the robot itself has the ability to determine the actions to be taken to perform a task, using a perception system that helps it. It also needs a cognition unit or a control system to coordinate all the subsystems that comprise the robot. The basics of mobile robotics consist of the fields of locomotion, perception, cognition, and navigation.

The term “robot” generally implies some anthropomorphic (human-like) appearance. The tendency to think about robots as having a human-like appearance may stem from the origins of term “robot”. The word “robot” came into the popular consciousness on January 25, 1921, in Prague with the first performance of karel capek’s play, R.U.R (Rossum’s Universal robots). In R.U.R, an unseen inventor, Rossum, has created a race of workers made from a vat of biological parts, smart enough to replace human in any job. The term “robot” (robota) was used for the first time capek’s play R.U.R and means slave servant or forced labour.

Mobile robots are robots that can move from one place to another autonomously, that is without assistance of external human operators. Unlike the majority of industrial robots that can move only in a specific workspace, basically they operate from a stationary and fixed position and have a bounded operating range. Industrial robots are automated, programmable and capable of movement about three or more axes. Typical applications of robots include welding, painting, assembly, pick and place for printed circuit boards, packaging and labeling, palletizing, product inspection, and testing; all accomplished with high endurance, speed, and precision. They can assist in material handling.

CLASSIFICATION OF MOBILE ROBOTS

- **Based on mechanical configuration:**

Stationary robot (arm/manipulator)

Manipulators and industrial robots are examples of this type. The robot’s base is fixed and they consist of an open kinematic chain, mainly with an end-effector with specials tools which not only handle objects but can also perform tasks such as welding, painting, assembling, machining, and so on. Robots of this type include Abb, Kuka, Fanuc, Staubli, Kawasaki, Comau, Wittman, and so on. Other important stationary robotic systems are grasping devices. Grasping is an important part of handling and from the beginning grasping devices were conceived mainly to help humans with handling tasks, providing solutions that can be classified into two categories: tools and prostheses. Over time, grasping devices have come to be used in many sectors, such as industry; agriculture, and so on, and a variety of robotic hand and finger mechanics have been developed. A broad overview can be found in Carlos et al

<u>Land-based robots</u>

a) Wheeled mobile robots

Wheels are one of the most important systems for robot locomotion, and autonomous intelligent vehicles (AIVs) are part of a challenging research field in mobile robotics, which relies on principles such as pattern recognition and signal–image processing. They will play an important role in transport, logistics, and distribution. The use of wheels is simpler than using treads or legs and is easier to design, build, and program when the robot is moving on flat, nonrugged terrain. They also tend to be much cheaper than their legged counterparts. Wheel control is less complex and they cause less wear and tear on the surface where they move in comparison with other solutions. Another advantage is that they do not present any great difficulty in terms of balance issues, since the robot is usually in contact with a surface. The main disadvantage of wheels is that they are not very good at navigating over obstacles, such as rocky terrain, sharp surfaces, or areas with low friction.

b) Walking or legged mobile robots

Legs are another common form of locomotion, giving rise to walking robots. Although they are usually more expensive than wheels, legs have several advantages over wheels. The greatest advantage is their transversality and efficiency and the fact that they can also move on soft and uneven terrain, better mobility, better energy efficiency, better stability, and a smaller impact on the ground. Walking robots also have the advantage of easily coping with obstacles or cracks found in the environment; in short, adaptability and maneuverability on rough terrain. There are many types of walking robots depending on the number of legs. Among the most important are biped (humanoids), four-legged (quadruped), six-legged, and so on.

Air-based robots

An unmanned aerial robot commonly known as “drone” is a machine that performs a Pre programmed task with or without human interaction and it is inspired by an airplane’s operation. The most advanced ones can now take off and land completely independently of the actions of their operators. Initially they were mostly used in military applications but they expanded rapidly to other applications such as scientific, agricultural, commercial, recreational, policing, and surveillance, product deliveries, distribution and logistics, aerial photography, and so on .

Water-based robots:

One of man’s oldest goals has been to explore the oceans and underwater areas that are inaccessible to him. As an important branch of mobile robots, the underwater vehicle manipulator system is one the hottest research topics nowadays. Many devices have been built for this, including robotic systems. Ocean One is an example of a submarine robot. It is a humanoid robot that explores the seabed. It takes advantage of the best of remotely operated vehicles and the advantages of humanoid robots, such as having a robotic hand with which to rescue objects as if it were a human being.

- ### Based on control system:

Non autonomous guided mobile robots:

Guided mobile robots or non-autonomous mobile robots require some sort of guidance system or instruction to make a movement that allows them to travel pre-defined navigation maps in a controlled environment. The pre-defined navigation map such as magnetic tape, bar codes, wire or sensors installed on the environment's floor that creating an inflexible environment. These are the following types:

1. **Autonomous Guided Vehicle (AGV):** This AGV requires the external guidance system in the form of magnetic strips to travel. These follow a rigid form of the preset route. Typical AGV applications incorporate transportation of raw materials, work-in-progress, and finished goods in support of manufacturing production lines, and storage/retrieval or other movements in support of picking in warehousing and distribution applications. AGVs provide automated materialmovement for a variety of industries including Automotive, Food & Beverage, Chemical, Hospitals, Manufacturing, Pharmaceutical, Paper.

2. Rail Guided Vehicle/Cart (RGV/RGC): RGV/RGC is a fast, flexible and easily installed material transport system that travels at a predefined path guided by rails or tracks. RGC has separate input/output stations that allow it to perform multiple operations at once. These mobile robots are an efficient, cost-effective and fast option for complex sorting applications.

3. Guided Fork-lifts: This specific AGV type is inspired by the conventional human manned forklifts. These forklifts are becoming increasingly complex and intelligent full of autonomy for some applications. These could manned/unmanned traveling with the help of external devices such as tablets, human, etc. The forklift AGV is designed to provide both horizontal and vertical movement of the load.

Autonomous guided mobile robots:

Autonomous mobile robots (AMR) are just like humans; can make their own decisions and then perform tasks accordingly. Autonomous robots can perceive their environment and remember it.Based on this info they navigate in a controlled environment without any predefined path or electro-magnetic guidance map, that way they offer flexibility to a large extent. AMRs also optimize the travel distance by calculating the shortest path for every mission & drive efficiency in the warehouse.

Let's look into a few of its applications:

1.AMR for Good-to-picking: This includes robots bringing mobile shelf units filled with items to a workstation. In this case, pickers remain at their workstations while software-driven AMRs deliver shelves with different materials directly to the order pickers' workstation.

2. Picking Assist Autonomous Mobile Robots: In this case, the robots travel to pick locations, where operators deliver ("pick") goods based on the robot's needs. They are an AMR base with an

operator interface that provides information about picking order. The robot tells the operator "I want this item and here is where you can find it". The user interface is also interactive, being possible to provide further info about the product or receiving info from the operator such as "picking accomplished".

3.Unmanned Aerial Vehicles (UAVs): These are basically drones moving large products through the air in distribution centers with the help of RFID-scanning technology to offer real-time inventory visibility in the warehouse. Guided autonomously by remote control, UAVs can sense their environment and navigate on their own.

4.Sorting Robots: These robots play an important role in high speed sorting esp in fulfilment centres. These robots work on a mezzanine with chutes/rabbit holes for location or order positions. Sortation is easily achieved by utilizing a fleet of sorting robots that sort the orders by dumping them through chutes/rabbit holes. The dropped orders or parcels are collected in sacks, gaylords or containers, which will be shipped directly to us.

1.2 PATH PLANNING FOR MOBILE ROBOT

The major emphasis in the field of autonomous control is the need for path planning. There is increased attention in the scientific community to enhance the knowledge of automation systems for different applications, such as chemically polluted or harmful locations. Robotic path planning is an appealing research study in the field of robotics . Since mobile robotics are used in a vast array of applications, numerous researchers have been working on various approaches in order to conquer a few of the significant challenges faced in autonomous navigation. These challenges restrict its usage in many applications, including industrial and military fields robot path planning is the process of finding an enhanced collision-free path from a start to a predefined goal point through a certain given cluttered real world environment within the shortest possible time. Mobile robot path planning has a few main properties according to type of environment, algorithm and completeness. The properties are whether it is static or dynamic, local or global and complete or heuristic. The **static path planning** refers to environment which contains no moving objects or obstacles other than a navigating robot and **dynamic path planning** refers to environment which contains dynamic moving and changing objects or obstacles. If **the path planning is global (off-line)** , information about the environment is already known based of map, cells, grid or etc and if the **path planning is local (on-line)**, the robot has no information about the environment and robot has to sense the environment before it decides to move for obstacle avoidance and generate trajectory planning toward target [1]. Local path planning methods use ultrasonic sensors, laser range finders, and on-board vision systems to perceive the environment to perform an on-line planning. In our paper, the workspace for the navigation of the mobile robot is assumed to be unknown and it has stationary obstacles only[2].

In this present work, determination of optimal path from initial to target point by considering online local path planning for a mobile robot system has been consider. The robot has the task to reach goal avoiding collision between robot and obstacle based on the modified PointBug algorithm using LASER range sensor.

1.3 LITERATURE SURVEY:

There are many work and researches have been done in the field of mobile robot path planning in last many years. Some of these research and works have been reported below.

Navid Toufan· Aliakbar Niknafs [1] proposes a new algorithm Robot path planning based on laser range finder and novel objective functions in grey wolf optimizer. In their research, the distributed multi-robot path planning problem is mainly done in three stages. All stages which are introduced here, operate for each robot individually. Firstly, the laser range finder (LRF) sensor, which is mounted on the top of robots, attempts to sense the environment based on sensing range limitation. The sensing is operated one time in each step before the robot starts to move. In the second stage, the current position of the robot and the sensor output are considered as the inputs for the procedure of the IGWO (IPSO) algorithms. Further, the multi-objective function is proposed in IGWO (IPSO) algorithms to evaluate the candidate solutions and enable them to converge to the optimal solution. The last stage directs the robot to the best position that has been founded in the second stage and updates the robot's current position.

Aisha Muhammad , Mohammed A. H. Ali ,* , Sherzod Turaev , Rawad Abdulghafor , Ibrahim Haruna Shanono , Zaid Alzaid , Abdulrahman Alruban , Rana Alabdan , Ashit Kumar Dutta and Sultan Almotairi ,* [2] develop a new mobile robot path planning algorithm, called generalized laser simulator (GLS), for navigating autonomously mobile robots in the presence of static and dynamic obstacles. This algorithm enables a mobile robot to identify a feasible path while finding the target and avoiding obstacles while moving in complex regions. An optimal path between the start and target point is found by forming a wave of points in all directions towards the target position considering target minimum and border maximum distance principles. The algorithm will select the minimum path from the candidate points to target while avoiding obstacles. The obstacle borders are regarded as the environment's borders for static obstacle avoidance. However, once dynamic obstacles appear in front of the GLS waves, the system detects them as new dynamic obstacle borders

James Ng & Thomas Braunl [3] explained that the Bug algorithm family is well-known robot navigation algorithms with proven termination conditions for unknown environments. Eleven variations of Bug algorithm have been implemented and compared against each other on the Eye Simsimulation platform and discussed their relative performance for a number of different environment types as well as practical implementation issues.

Zohaib, M., Pasha, M., Riaz, R.A., Javaid, N., Ilahi, M., & Khan, R.D [4] explained Obstacle avoidance is an important task in the field of robotics, since the goal of autonomous robot is to reach the destination without collision. Several algorithms have been proposed for obstacle avoidance, having drawbacks and benefits. In this survey paper, they mainly discussed different algorithms for robot navigation with obstacle avoidance. They also compared all provided algorithms and mentioned their characteristics; advantages and disadvantages, so that they can select final efficient algorithm by fusing discussed algorithms. Comparison table is provided for justifying the area of interest.

Muhammad Zohaib, Syed Mustafa Pasha, Nadeem Javaid, Jamshed Iqbal [5] proposed an intelligent obstacle avoidance algorithm to navigate an autonomous mobile robot. The presented Intelligent Bug Algorithm (IBA) over performs and reaches the goal in relatively less time as compared to existing Bug algorithms. The improved algorithm offers a goal oriented strategy by following smooth and short trajectory. This has been achieved by continuously considering the goal position during obstacle avoidance. The proposed algorithm is computationally inexpensive and easy to tune. The paper also presents the performance comparison of IBA and reported Bug algorithms. Simulation results of robot navigation in an environment with obstacles demonstrate the performance of the improved algorithm.

Lee, Donghyun, et al [6] presented in a modified path planning method based on the APF (artificial potential field) is proposed for the mobile robot navigation system. The path planning method is very important to the robot navigation system. The APF is a commonly used path planning method because of its advantages of simple processing and online realization in the unknown environment. However, the T-APF (traditional-APF) has shortcomings such as local minima and path inefficiency problems. To overcome these drawbacks, the NP-APF (new point-APF) is proposed. NP-APF is a specialized method for using the LiDAR. The key idea of the NP-APF is that it creates the new point of the attractive force what can improve performance of the APF by solving its drawbacks. The new point is created when the obstacles block a path which makes straight-line from the mobile robot to the goal, and it helps the mobile robot overcome local minima and path inefficiency problems. The new point is created at the most reasonable place where no obstacle is detected by the LiDAR. The simulation results show the improvement of performance compared with the T-APF

Dib, Lynda [7] presented a novel sensor-based path-planning algorithm called "E-Bug" (Euclidian Bug); since it uses the Euclidian distance to choose the shortest path. In addition, they expose and discuss many deficiencies in the searching process of almost Bug path-planning algorithms, as well as many routing algorithms in wireless networks and others. These deficiencies are illustrated by cons examples. To overcome and solve these limits we propose a new formalism, based on sudden point concept, on which the new proposed algorithm E-Bug is based. The validity and the performance of our algorithm are demonstrated by a huge amount of simulations results on randomly generated environments. The robustness of E-Bug is proved comparing it, using the same conditions, with other powerful Bug's algorithms (PointBug, TangentBug, K-Bug and other). Moreover, they compare E-Bug with some algorithms that have a full prior knowledge of the robot's environment.

Mohamad, Z [8] proposed the Bug algorithm is a local path planning methodology which detects the nearest obstacle as a mobile robot moves towards a target with limited information about the environment. It uses obstacle border as guidance toward the target. In Bug algorithm, the robot circumnavigates the obstacle till it finds certain condition to fulfill algorithm criteria to leave the obstacle toward target point. This paper introduces an approach utilizing a new algorithm called PointBug that attempts to minimize the use of outer perimeter of an obstacle (obstacle border) by looking for a few important points on the outer perimeter of obstacle area as a turning point to target and finally generates a complete path from source to target. The less use of outer perimeter of obstacle area produces shorter total path length taken by a mobile robot. This approach is then compared with other existing selected local path planning algorithm for total distance and a guarantee to reach the target.

Leena.N , K.K.Saju [9] reviewed and investigated the different path planning algorithms and techniques for mobile robot navigation. It describes the various developments and techniques that have been applied for navigation of robots in static and dynamic environments with special focus on the soft computing approaches.

Jun-Hao Liang, Ching-Hung Lee [10] proposed a novel design approach for on-line path planning of the multiple mobile robots system with free collision. Based on the artificial bee colony (ABC) algorithm, we propose an efficient artificial bee colony (EABC) algorithm for solving the on-line path planning of multiple mobile robots by choosing the proper objective function for target, obstacles, and robots collision avoidance. The proposed EABC algorithm 15 | P a g e enhances the performance by using elite individuals for preserving good evolution, the solution sharing provides a proper direction for searching, and the instant update strategy provides the newest information of solution. By the proposed approach, the next positions of each robot are designed. Thus, the mobiles robots can travel to the designed targets without collision. Finally, simulation results of illustration examples are introduced to show the effectiveness and performance of the proposed approach.

C. Balaguer, A. Martí [11] presents a new collision-free path planning algorithm for mobile sensor-based robot which moves in unknown static environment. The 3D environment is a closed room formed by its borders and by the obstacles of straight prismatic type. The knowledge of the environment is augmented using real-time data from on-board robot's sensors: stereo vision and distance. The free-space is represented by the triangular corridors, which formed a map of the environment. The collision-free path planning algorithm searches the optimum trajectory in the map tree. The optimization criteria are: length of the trajectory and number of high consumption stereo image processing operations. The results of the algorithm's simulation make it able for implementation in real robot..

Farouk MEDDAH, Lynda DIB [12] presents P* “P-Star” which is a new algorithm for sensor based path planning. The algorithm is based on Point-Bug algorithm where some improvements and modifications to overcome some important problems (like infinite loops and the bypass of some sub-paths). Moreover, some simulation and comparisons with PointBug are done to evaluate and to verify the performance and the power of the proposed algorithm.

Hoc Thai Nguyen, Hai Xuan Le [13] developed and implemented a new path planning method for mobile robots (MR). On the other hand based on the shortest path from the start to goal point, this path planner can choose the best moving directions to reach the target point as soon as possible. On the other hand, with intelligent obstacles avoidance, the method can find the target with near shortest path length while avoiding some infinite loop traps of several obstacles in unknown environments. The combination of two approaches helps the mobile robot to reach the target with very reliable algorithm. Effectiveness of the algorithm is justified using simulation software in both static and dynamic environments.

N. Nirmal Singh, Avishek Chatterjee, Amitava Chatterjee, Anjan Rakshit [14] describes the real-life implementation of a mobile robot navigation scheme where vision sensing is employed as primary sensor for path planning and IR sensors are employed as secondary sensors for actual navigation of the mobile robot with obstacle avoidance capability in a static or dynamic indoor environment. This two-layer based, goal-driven architecture utilizes a wireless camera in the first layer to acquire image and perform image processing, online, to determine sub-goal, employing a shortest path algorithm, online. The sub-goal information is then utilized in the second layer to navigate the robot utilizing IR sensors. Once the sub-goal is reached, vision based path planning and IR guided navigation is reactivated. This sequential process is continued in an iterative fashion until the robot reaches the goal. The algorithm has been effectively tested for several real-life environments created in our laboratory and the results are found to be satisfactory.

Zi-Xing CAI, Zhi-Qiang WEN, Xiao-bing ZOU, Bai-fan CHEN [15] presents an inverse D* algorithm for path-planning under unknown environments. In this inverse D* algorithm, the local potential energy around the current position is created firstly by defining the robot distance, then the leave point is searched to be regarded as the local goal position satisfying requirement of the rolling optimization. The leave point is searched locally and iteratively until the robot reaches the goal finally. The experimental results show the validity of the algorithm.

Yang-Ge Wu, Jing-Yu Yang, Ke Liu [16] presents a multi-sensor integrated vision system and sensor fusion algorithm for the navigation of an autonomous mobile robot equipped with laser range finder radar (LRFR) and a color CCD camera to acquire information about the environment. The 2D model of the environment is constructed and the obstacles on the road are detected by fusing knowledge included in the range of images obtained by the LRFR and the color camera.

James Ng [17] compares and analyses the practical aspects of path-planning and navigation algorithms for autonomous robots. The algorithms bug1, bug2, alg1, alg2, distbug, tangentbug and D* were implemented and simulated on simulation software.

Jang Gyu Lee, Hakyoung Chung [18] presents a new methodology for global path planning for an autonomous mobile robot in a grid-type world model. The value of a certainty grid representing the existence of an obstacle in the grid is calculated from readings of sonar sensors. In the calculation, a way of utilizing three sonar sensors readings at a time is introduced, resulting in more accurate world model. Once the world model is obtained, a network for path planning is built by using the model. The global paths, defined as the shortest paths between all pairs of nodes in the network, are calculated. A fast algorithm using a decomposition technique is proposed for real-time calculation. The new methodology has been implemented on the mobile robot whose role is to transport materials in flexible manufacturing system. The results show that the proposed method of certainty grids satisfactorily represents a precise environment, including the location of obstacles. Thus, the robot successfully comprehends its surroundings, and navigates to its destinations along optimal paths.

Takanori Shibata, Toshio Fukuda, Kazuhiro Kosuge, Fumihito Arai [19] proposed a new hierarchical strategy for path-planning of multiple mobile robots using Genetic Algorithms (GAs). When a mobile robot moves from a point to a goal point, it is necessary to plan an optimal or feasible path avoiding obstructions and minimizing a cost such as time, energy, and distance. This planning is referred to as the selfish-planning. When many robots move in a same space, it is necessary for each robot to select the most reasonable path so as to avoid collisions with other robots and to minimize the cost. This planning is referred to as the coordinative-planning. The GAs are applied hierarchically to both planning's of multiple mobile robots.

Zhao-Qing Ma, Zeng Ren Yuan [20] developed and implemented a real—time navigation and obstacle avoidance method based on grids on the mobile robot THMR-2. This method permits the detection of unknown obstacles and avoidance collision based on the information of ultrasonic sensors while autonomously steering the mobile robot toward the given target in smooth and continuous motion. Experimental results are given in some typical environment. The strong power of the method has been demonstrated.

B. Margaret Devi, Prabakar S [21] proposed an algorithm named dynamic point bug which can be included in bug algorithm family. The main task of a robot is to search a collision free path in order to reach the target specified. The main problem in robot navigation is localization i.e. the robot should know its present location. Here in this algorithm the localization problem is solved by using graphical method. Dynamic point bug algorithm has been implemented for robot navigation system. This proposed algorithm provides a solution to identify the present location of the robot while moving towards target based on coordinates estimation.

Lei Cai, Juanjuan Yang, Li Zhao, Lan wu [22] shows The quadratic programming problem has broad applications in mobile robot path planning. This article presents an efficient optimization algorithm for globally solving the quadratic programming problem. By utilizing the convexity of univariate quadratic functions, we construct the linear relaxation programming problem of the quadratic programming problem, which can be embedded within a branch-and-bound structure without introducing new variables and constraints. In addition, a new pruning technique is inserted into the branch-and-bound framework for improving the speed of the algorithm. The global convergence of the proposed algorithm is proved. Compared with some known algorithms, numerical experiment not only demonstrates the higher computational efficiency of the proposed algorithm but also proves that the proposed algorithm is an efficient approach to solve the problems of path planning for the mobile robot

Lei Cai, Juanjuan Yang, Li Zhao, Lan wu [23] represented a novel detection algorithm for vision systems has been proposed based on combined fuzzy image processing and bacterial algorithm. This combination aims to increase the detection efficiency and reduce the computational time. In addition, the proposed algorithm has been tested through real-time robot navigation system, where it has been applied to detect the robot and obstacles in unstructured environment and generate 2D maps. These maps contain the starting and destination points in addition to current positions of the robot and obstacles. Moreover, the genetic algorithm (GA) has been modified and applied to produce time-based trajectory for the optimal path. It is based on proposing and enhancing the searching ability of the robot to move towards the optimal path solution. Many scenarios have been adopted in indoor environment to verify the capability of the new algorithm in terms of detection efficiency and computational time.

Valencia R., Andrade-Cetto J. [24] proposed in this Chapter that Pose SLAM graphs can be directly used as belief roadmaps and thus used for path planning under uncertainty. The method they present in this Chapter devises optimal navigation strategies by searching for the path in the pose graph with the lowest accumulated robot pose uncertainty, i.e., the most reliable path to the goal.

Li G, Chou W [25] described as a challenging optimization problem, path planning for mobile robot refers to searching an optimal or near-optimal path under different types of constrains in complex environments. In this paper, a self-adaptive learning particle swarm optimization (SLPSO) with different learning strategies is proposed to address this problem. First, we transform the path planning problem into a minimisation multi-objective optimization problem and formulate the objective function by considering three objectives: path length, collision risk degree and smoothness. Then, a novel self-adaptive learning mechanism is developed to adaptively select the most suitable search strategies at different stages of the optimization process, which can improve the search ability of particle swarm optimization (PSO). Moreover, in order to enhance the feasibility of the generated paths, we further apply the new bound violation handling schemes to restrict the velocity and position of each particle. Finally, experiments respectively with a simulated robot and a real robot are conducted and the results demonstrate the feasibility and effectiveness of SLPSO in solving mobile robot path planning problem

Haj Darwish, Ahmed, Abdulkader Joukhadar, and Mariam Kashkash [26] presents a solution to plan a path using a new form of the Bees Algorithm for a 2-Wheeled Differential Drive mobile robot. This robot is used in an indoor environment. The environment consists of static and dynamic obstacles which are represented by a continuous configuration space as an occupancy map-based. The proposed method is run in two respective stages. Firstly, the optimal path is obtained in the static environment using either the basic form or the new form of the Bees Algorithm. The initial population in the new form of the Bees Algorithm consists only of feasible paths. Secondly, this optimal path is updated online to avoid collision with dynamic obstacles. A modified form of the local search is used to avoid collision with dynamic obstacles and to maintain optimality of sub-paths. A set of benchmark maps were used to simulate and evaluate the proposed algorithm. The results obtained were compared with those of the other algorithms for different sets of continuous maps. This comparison shows the superiority of the new form of the Bees Algorithm in solving this type of the problems. The proposed method was also tested using AmigoBot robot. In this experiment, the proposed method was implemented using multi-threading techniques to guarantee real time performance at the dynamic stage. The results of this experiment prove the efficiency of the proposed method in a real time.

Akka, Khaled, and Farid Khaber [27] expressed Ant colony algorithm is an intelligent optimization algorithm that is widely used in path planning for mobile robot due to its advantages, such as good feedback information, strong robustness and better distributed computing. However, it has some problems such as the slow convergence and the prematurity. This article introduces an improved ant colony algorithm that uses a stimulating probability to help the ant in its selection of the next grid and employs new heuristic information based on the principle of unlimited step length to expand the vision field and to increase the visibility accuracy; and also the improved algorithm adopts new pheromone updating rule and dynamic adjustment of the evaporation rate to accelerate the convergence speed and to enlarge the search space. Simulation results prove that the proposed algorithm overcomes the shortcomings of the conventional algorithms.

Roy, Nirmalya, et al [28] proposed Development and path planning in mobile robots is an exigent field of robotics. The objective of this paper is to show the use of Q-learning for navigation in indoor environments. Planning the shortest path from current state to the goal state using images captured from ceiling of the indoor environment. Captured image is tried to processed through different image processing and machine learning techniques. Obstacles in the path of the robot is also tried to processed by calculating Adaptive Gaussian Thresholding of the image captured. Position of the robot is tried to be tracked using template matching of CV. Q-learning techniques are applied to plan path of the mobile robot from current start to the goal state.

Kuisong Zheng , Feng Wu and Xiaoping Chen [29] proposed a paper describes the development of a laser-based people detection and obstacle avoidance algorithm for a differential-drive robot, which is used for transporting materials along a reference path in hospital domains. Detecting humans from laser data is an important functionality for the safety of navigation in the shared workspace with people. Nevertheless, traditional methods normally utilize machine learning techniques on hand-crafted geometrical features extracted from individual clusters. Moreover, the datasets used to train the models are usually small and need to manually label every laser scan, increasing the difficulty and cost of deploying people detection algorithms in new environments. To tackle these problems, (1) they propose a novel deep learning-based method, which uses the deep neural network in a sliding window fashion to effectively classify every single point of a laser scan. (2) To increase the speed of inference without losing performance, we use a jump distance clustering method to decrease the number of points needed to be evaluated. (3) To reduce the workload of labeling data, we also propose an approach to automatically annotate datasets collected in real scenarios. In general, the proposed approach runs in real-time and performs much better than traditional methods. Secondly, conventional pure reactive obstacle avoidance algorithms can produce inefficient and oscillatory behaviors in dynamic environments, making pedestrians confused and possibly leading to dangerous reactions. To improve the legibility and naturalness of obstacle avoidance in human crowded environments, we introduce a sampling-based local path planner, similar to the method used in autonomous driving cars. The key idea is to avoid obstacles by switching lanes. We also adopt a simple rule to decrease the number of unnecessary deviations from the reference path. Experiments carried out in real-world environments confirmed the effectiveness of the proposed algorithms

1.4 OBJECT AND SCOPE OF PRESENT RESEARCH WORK

Aim of the present work is to develop necessary algorithm to navigate a mobile robot in presence of obstacles from a starting point to a destination point using sensory feedback. A Parallax mobile robot (ActivityBot), purchased in the Robotics laboratory of Production Engineering Department of Jadavpur University, has been used for the present project. It is then necessary to obtain a path for avoiding obstacles present in static environments in the workspace. After location of obstacles is determined by LASER range sensor, the ActivityBot has moved towards the destination from the starting position following the path using Modified PointBug algorithm, a simple local path planning algorithm.

The algorithm determines the next point to move for the robot forward target from any current point, which is the starting point at the beginning. Then next point is always determined on the basis of the output of an LASER range sensor fitted to the mobile robot that gives the distance of nearest obstacle from the sensor.

Hence the main objectives of the present work are as follows:

- To set up an arrangement for the workspace consisting of an Activity-Bot mobile robot with LASER range sensor and obstacles on a suitable worktable, having a marked boundary.

- To mount the LASER range sensor on the Activity-Bot mobile robot system, and make necessary hardware connection to connect it to the Propeller Chip microcontroller through its input-output port (pins). Arrangement has been made for rotation of the LASER range sensor through 180^o by a servo motor with reduction gear drive fitted to the ActivityBot.

- To develop a simplified algorithm based on Point-Bug algorithm and to modify it for the purpose of moving a mobile robot of particular dimensions towards the goal avoiding static obstacles.

- To develop a program in Propeller-C language for producing necessary movements of the Activity-Bot mobile robot in presence of static obstacles for moving from a starting point to a target point using Modified PointBug Algorithm.

- To run the program for different layout of workspace for testing the algorithm.

<u>Scope</u>

The design, analysis, algorithm, techniques described in the present thesis will be applicable to any mobile robot with two wheels driven by separate motors. However the program and the experimental results will be applicable only for the specific mobile robot used in the present project.

CHAPTER-2

2.0 DIFFERENT PATH PLANNING TECHNIQUES FOR MOBILE ROBOT

2.1 BUG ALGORITHMS

Bug algorithms are fundamental and complete algorithms with provable guarantees, since they let the robot to reach its destination if it lies in given space. Bug algorithms solve the navigation problem by storing only a minimal number of way points, but without generating a full map of the environment. In these algorithms the robot takes an action on the basis of current percept of sensor without taking into account the previous path and actions. It has two behaviours, one is ‘‘move towards goal’’ and another is ‘’obstacle avoidance’’. In obstacle avoidance, it just avoids obstacle by following its edges and then restarts to move toward goal without considering any other parameter

Navigation in an unknown 2D environment is one of the standard problems in robotics. A mobile robot is being placed at a starting position (S) and has to find its way to a goal position (G) that is specified by distance and direction relative to the starting position; no other information about the environment is known to the robot. The robot only has to reach the target position or terminate if the target is unreachable if it does not have to map its environment. Therefore, a particular navigation algorithm can have a statistically better performance than another, but may not be better for any possible environment setting [3].

The Bug model makes following assumptions about the robot.

1. The robot is a point object.

2. Navigation plane will be two-dimensional.

3. No of obstacle is finite and have placed within a boundary region.

4. location and shape about obstacle is completely unknown to robot.

5. Robot can sense its position and can measure its travelled distance.

6. Robot has computation power and memory to direct towards goal and can memorise the distance between two points.

The following algorithms from the Bug family have been implemented and evaluated: Bug1, Bug2, Alg1, Alg2, DistBug, IBA, Class1, Rev1, Rev2, OneBug, LeaveBug and TangentBug. It is to be noted that this is not a complete list of Bug algorithms in existence. There are more, e.g. VisBug-21, VisBug-22, HD-1, Ave, RoverBug, WedgeBug and CautiousBug. Those were not included for brevity and because they are quite similar to some of the included algorithms. For instance, RoverBug, WedgeBug and CautiousBug are similar to TangentBug.In this paper we have discussed few of important and fundamental Bug algorithm like Bug1, Bug2, DistBug, IBA, Tangent Bug algorithm.

2.1.1 Bug-1 Algorithm :

In this algorithm [4] when robot detects an obstacle it starts moving around it until reaches to starting point from where it has started moving around the obstacle. During its movement around an obstacle, it calculates a leaving point with minimum distance to destination and generates new path from calculated leaving point to destination. After its one complete revolution around obstacle, it restarts its motion around obstacles until reaches to leaving point and starts moving on new generated path to reach the destination. Fig-2.1 shows the trajectory of robot under Bug-1 algorithm. Here, leaving point is **(x_1,y_1)** after detecting and avoiding the obstacle, so robot calculate new path from leaving to destination**(x_2,y_2)** using straight line equation. So the line equation is **$y_1=m*x+c$;** where '**c**' is the y axis intercept and '**m**' [**$m=\tan^{-1}\{(y_2-y_1)/(x_2-x_1)\}$**] is the slope [4].

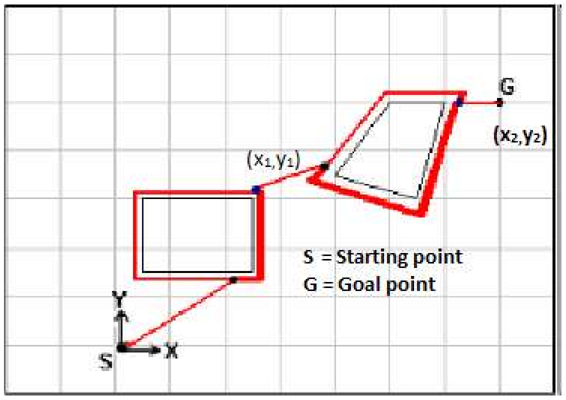


Fig-2.1 : Obstacle avoidance with Bug algorithms (Trajectory of Bug-1 algorithm)

Robot follow this line and move toward goal untill a new obstacle is detected. One disadvantage of Bug-1 algorithm is, when the robot is following the edge of obstacle 1, it may collide with a neighbouring obstacle 2 in case when the later is in very close proximity to the first obstacle or the gap between them is less than the width of the robot.

2.1.2 Bug-2 Algorithm:

Bug-2 algorithm [4] generates slope [**m; m=tan^{-1}\{(y_2-y_1)/(x_2-x_1)\}**] from an initial position S (x_1,y_1) to destination G (x_2,y_2) and robot starts following it until it interrupted by obstacle. When it interrupted, it follows the edge of obstacle and calculates new slope from every new position until the new slope becomes equal to the original slope. After reaching on point having same slope as previous, it starts moving to destination by following pervious generated path. Bug-2 algorithm is shown below in fig-2.2.

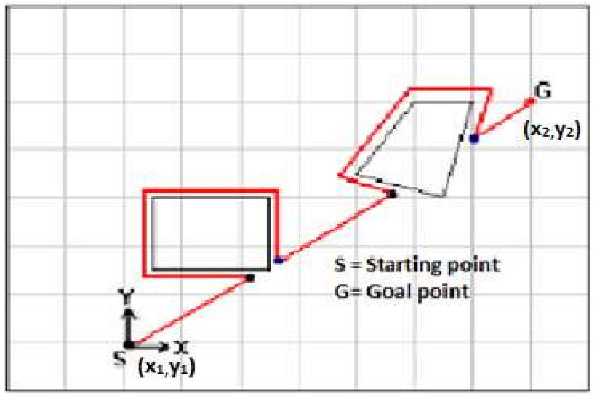


Fig-2.2: Obstacle avoidance with Bug algorithms (Trajectory of Bug-2 algorithm)

2.1.3 Dist-Bug Algorithm:

This algorithm is based on distance, in which robot moves from source to destination on path having minimum distance. When robot faces an obstacle in path, it starts following the edge of obstacle simultaneously; it calculates the distance of destination from each point. The point with the minimum distance is known as leaving point. When it finds the leaving point during its motion around an obstacle, it generates a new path and starts following it until reaches to destination [4]. as shown in the fig-2.3.

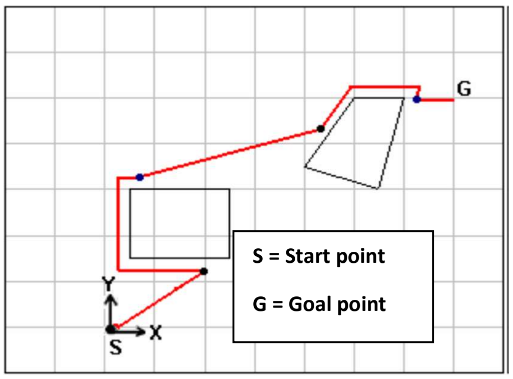


Fig-2.3: Obstacle avoidance with Bug algorithms (Trajectory of Dist-Bug algorithm)

2.1.4 Intelligent Bug Algorithm (IBA):

In the category of Bug algorithms [5], Dist-Bug algorithm is most efficient as path cost is considered throughout the decision making process. However, it is not goal oriented and thus can take the robot far away from its goal position while avoiding obstacles. This is due to its leaving point decision during edge detection in obstacle avoidance behaviour since the goal information is not taken into account. This gives the clue to improve Dist-Bug algorithm by creating an approach to make it goal oriented and to take time to destination into consideration. Based on this, the proposed IBA offers an intelligent control to navigate the robot in maze environment.

The IBA algorithm has two fundamental steps one is ‘’move to goal’’ and another ‘’obstacle avoidance’’. The behaviours in IBA also depend on the present sensorial information of environment just as similar to the Dist-Bug i.e. whether obstacles are sensed or not. Initially, in move to goal behaviour, a reference path is generated from source to goal position and the robot is forced to follow it until an obstacle is encountered or destination is reached. The behaviour of the robot is changed to obstacle avoidance when an obstacle is sensed and the robot is commanded to follow the edges of the obstacle until leaving point is reached. In IBA, leaving point by taking the goal position into account is selected on the basis of free path toward the destination. The robot monitors the obstacles in the path towards destination while detecting edge in obstacle avoidance behaviour. Trajectory of IBA is shown in the fig-2.4 where ‘S’ is the start point and ‘G’ is the goal point.This condition, not introduced in Dist-Bug algorithm, offers goal orientation.

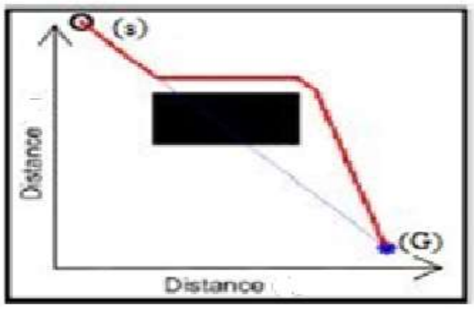


Fig-2.4: Trajectory of IBA

The condition dictates that in IBA, the leaving point is not taken on the basis of minimum distance to destination. The obstacle-free path towards goal is also considered. This ensures that the robot does not have to wait for the point having minimum distance to goal. The robot changes its behaviour to move to goal in order to generate new reference path, in case an obstacles-free path is sensed.

2.1.5 Tangent Bug Algorithm:

The TangentBug [6] algorithm was developed by Kamon, Rivlin and Rimon .Tangent Bug algorithm finds tangents to the obstacle and calculates distances of robot from points where they touch the obstacle. These points are denoted by pi, where i denote index number. In this approach robot takes path of the tangent which maximally decreases heuristic distance i.e. d(robot position, pi)+d(pi, goal). At times it has to act as bug algorithm by following the edge of the obstacle if heuristic distance starts increasing while moving towards pi. It is a memory type algorithm which exhibits motion to-goal and boundary following behavior. A value dmin which is the shortest distance observed so far between the sensed boundary of the obstacle and the goal and dleave which is the shortest distance between any point in the currently sensed environment and the goal are continuously updated. It terminates boundary following behavior when dleave < dmin. The figure 2.5 shows the path generated by tangent bug algorithm.

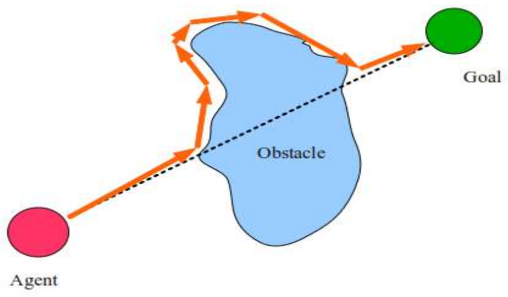


Fig- 2.5: Obstacle avoidance with Tangent Bug algorithms (Trajectory of Tangent-Bugalgorithm)

2.2 <u>POINTBUG ALGORITHM:</u>

PointBug, [7] recently developed navigates a point of robot in a 2D plane of unknown environment which is filled with stationary obstacles of any shape. This new algorithm introduces a new notion called ‘sudden point’. Sudden point is a point where a sudden change in distance of sensor’s range is detected. This algorithm determines where the next point to move toward target from a starting point. The next point is determined by output of range sensor which detects the sudden change in distance from sensor to the nearest obstacle. The sudden change of range sensor output is considered inconsistent reading of distance either it is increasing or decreasing. There are three possible changes (a) It can be from infinity to certain value (b) from certain value to infinity (c) from certain value to a certain value. The initial position of robot is facing straight to the target point and then the robot rotates left or right searching for sudden point. After the first sudden point is found, the rotation direction of the robot is according to position of straight line between current sudden point and target point or dmin line (The line that connects the current point to the target point). The rotation direction of robot is always toward position of dmin line. The value of dmin is the shortest distance in one straight line between sudden point and target point and its value is always recorded every time the robot reaches new sudden point. The robot always ignores the sensor reading at rotation of 1800 and its beyond to avoid detection of previous sudden point making the robot return to previous sudden point from its current point. If there is no sudden point found within a single 3600 rotation, the target is considered unavailable and the robot stops immediately.

This algorithm can also perform in dynamic environment by obtaining information from range sensor then Robot advances and adjusts its direction continuously to the target.

PointBug algorithm is suitable for an environment where there are not many obstacles and no spiral. The performance of the algorithm depends on total sudden points detected, less the sudden points detected better the performance.

Figure-2.6 shows a range sensor scanning a pentagon shaped obstacle from A to E with a graph showing the distance produced from range sensor in cm from A to E. The C line is perpendicular to the surface of obstacle which is the shortest distance detected to the obstacle. The value of distance increases constantly from C to B and from C to D. From point B to A from the graph, the value of distance is suddenly increased almost twice and from point D to E the value of distance is suddenly increased from a few centimetres to infinity. The point A and E are the sudden points and considered the points where the robot will move for the next point. Figure-2.7 shows the sudden points are detected on different shape of obstacles [8].

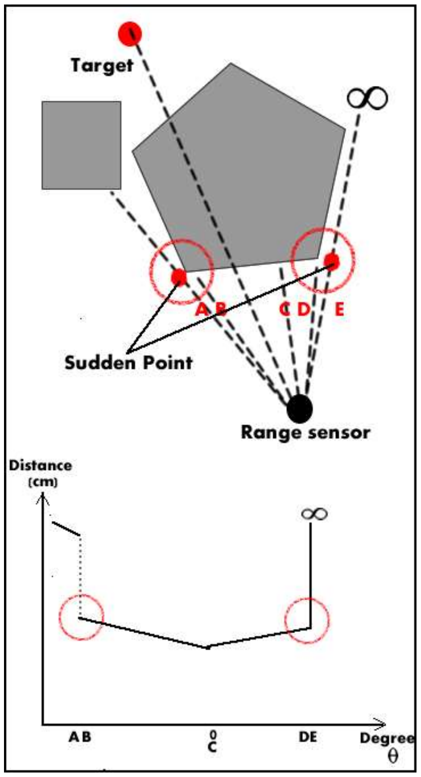


Fig-2.6: Range sensor is detecting an obstacle from left to right and right to left

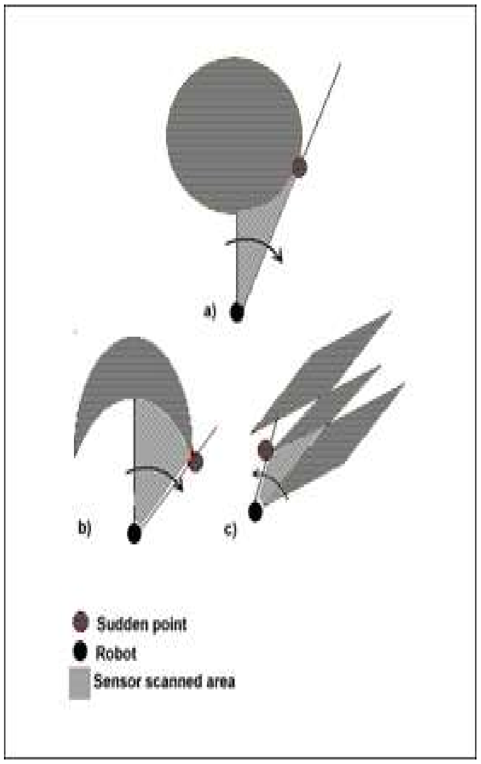


Fig-2.7: sudden point on different obstacle detecting by range sensor.

Figure-2.8 shows how the algorithm is working in an environment to solve local minima problem by detecting sudden points from a starting point to target point. The robot first faces the target point at the starting point and then rotates from point A until it finds a sudden point at point B. Robot then move to point B and at point B; it rotates to the right direction to find next sudden point because the dmin line is located right side of current robot direction and finds new sudden point at C. Robot rotates to the right again at point C and finds new sudden point at D. At point D, the robot still rotates to the right and finds last sudden point and stop at target point.

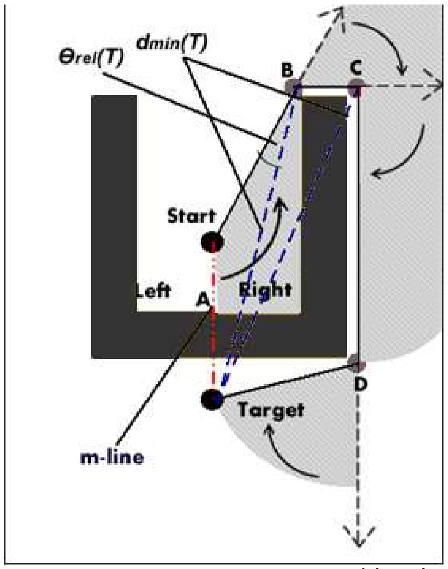


Fig.-2.8 : Trajectory generated by the PointBug to solve local minimal problem and shadowed area is scanned area.

Point of Movement	Sudden point Description
A to B	B (from certain value to infinity)
B to C	C (from infinity to certain value)
C to D	D (from infinity to certain value)
D to target	Infinity to target

Table 1: Explanation on how sudden point are found from starting point to Target from The fig.-2.8.

2.2.1 <u>PointBug Algorithm Analysis:</u>

The objective of PointBug algorithm [7] is to generate a continuous path from start point(S) to target point(T) where S,T are given. In this algorithm the distance between two point A and B is denoted by d(A,B) so d(S,T) indicate the distance between S to T which is constant; d(S,T)=D(constant). D(An, B) indicates that point A which is located at yth sudden point on the way to 'T' and 'P' is total length of connected sudden points from 'S' to 'T'. The line (S,T) is called the main line or m-line.

The path generated by connecting two consecutive sudden points will be straight line and total distance covered by the robot is calculated by summing all the straight line created with sudden points; so

$$P = \sum_{n=1}^{s+1}(\mathrm{A}_{n-1}, A_n) \text{............ (1)}$$

In PointBug algorithm, every sudden point found will produce a logical triangle which is formed from three points namely target point, current sudden point and previous sudden point. The line between target point and current sudden point is dmin line and its values are accumulated in an array starting from 0 which is distance from starting point and target point up to last sudden point before meeting target point. Value dmin[0] is assigned manually and it is the initial value required to run the algorithm. The values of dmin[1] to dmin[n] are obtained from cosines rule except dmin[0].

$$a^2 = b^2 + c^2 - 2bc \cos A \text{ .... (2)}$$

If a is dmin then;

$$\text{dmin} = \sqrt{b^2 + c^2 - 2bc\cos A} \text{............ (3)}$$

Here b is the distance between current sudden point to previous sudden point obtained from range sensor and c is the previous value of dmin. So the value of dmin[n] is

$$\text{Dmin}=\sqrt{b^2 + \mathrm{dmin}[\mathrm{n}-1]^2 - 2b\mathrm{dmin}[\mathrm{n}-1]\cos A} \text{............ (4)}$$

Here angle A is rotation between current direction to next direction if the robot located at starting position, otherwise;

$$\text{Angle A} = 180 - \text{Angle Adj} - \text{Angle Rot if Angle A} \leq 90 \text{ .... (5)}$$

Angle A = Angle Adj + Angle Rot if Angle A > 90 …………… (6)

In equation (5) and (6) Adj = Adjacent angle of triangle. Rot is rotational angle and angle Adj is calculated from sine rule; if sin B is Adj the

$$\text{Angle Adj} = \sin^{-1}\frac{\mathrm{b}\sin\mathrm{A}}{\mathrm{a}} \text{ …………… (7)}$$

where the b is previous dmin value and a is current dmin value.

Say If dmin[n]=0 then the robot is currently on target point. Now dmin[n] is the minimum distance between sudden point and target point. If its value is zero means the sudden point is on the target point, the value of angle A is zero and the value of previous dmin[n]is equal to distance between current sudden point and previous sudden point or c. Let's say value of previous dmin[n] is b, and from equation (3), the value of dmin[n] is;

$$\text{dmin[n]} = \sqrt{b^2 + b^2 - 2bb\cos A}$$

$$\text{dmin[n]} = \sqrt{2b^2 + 2b^2}$$

$$\text{dmin[n]} = 0$$

2.2.2 PointBug Algorithm Limitations

Problem Of Continuous Follow Of Advancing Direction :

The path generated by PointBug is not always optimum [7]. The continuous follow of the travel's direction leads robot to move far away from the optimal path. This PointBug limitation is illustrated by Figure -2.9 when the robot arrives to point (B), PointBug algorithm starts searching for the next sudden point. As this research starts from the current direction, the next sudden point will be the point (C) instead of the point (D) which is the best choice.

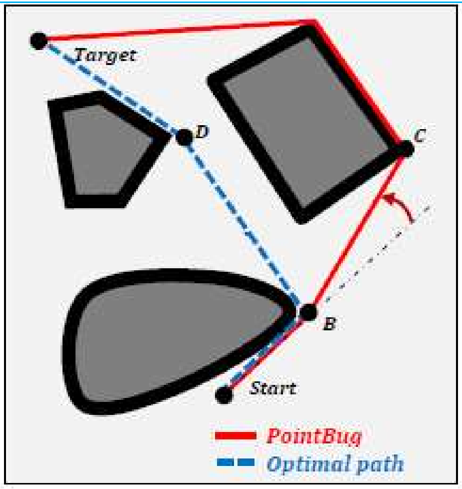


Fig-2.9: Problem still followed the direction of travel by PointBug algorithm.

Problem Of The Bypass Of Sub-Paths :

The searching strategy used by PointBug can produce another problem. Not only losing the optimal path, moreover the robot may never reach the target. An example illustrating this case is given by the Fig-2.10.

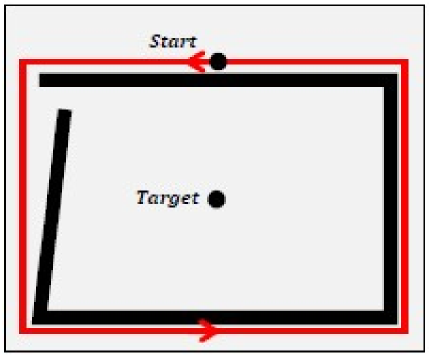


Figure-2.10: Example where PointBug algorithm can't reach the target.

In this example (Fig-2.10) PointBug algorithm can't reach the target in spite of its existence, furthermore, the robot continues moving in infinite loops.

The Choice of the Point Which Minimize the Angular Deviation Relative to Target Direction does not Produce the Optimal Path :

Many algorithms choose the next points to minimize the deviation angle from target direction. In this section we will demonstrate that even this principle is false:

PointBug algorithm uses this principle when searching the first sudden point. It starts from angle zero and increase it until finding the first sudden point, see Figure-3.1.

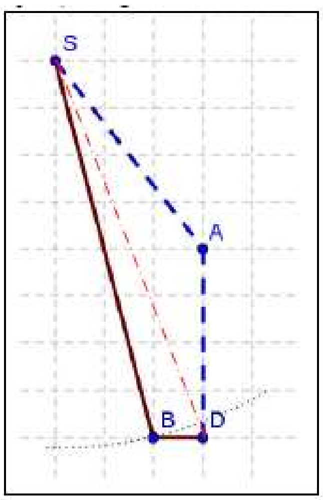


Fig-2.11: Example where minimizing the angular deviation does not produce the shortest path.

In this example (Figure 8) the deviation angle formed by the point (A) [SD, SA] is 24.4° and the path length (S)-(A)-(D) = 9cm, and the deviation angle formed by (B) [SD, SB] is 6° and the path length (S)-(B)-(D) = 9.246cm

- Optimal path 9 cm ; **SA =$\sqrt{4^2 + 3^2}$ =5** and **AD = 4** so **S-A-D = 9**
- PointBug (or other algorithm with the same principle)= **SB= $\sqrt{8^2 + 2^2}$ = $\sqrt{68}$ = 8.246** and **BD=1** so

  S-B-D = 9.246

Although the greater deviation angle formed by the point (A) [SD, SA] this path is shorter.

In the present work, the basic concept of considering the angle with minimum deviation is used in PointBug algorithm has been used with the following modifications:

a) Instead of considering the mobile robot as a ‘point’, the actual dimension of the mobile robot has been considered.
b) Whenever an obstacle is encountered in its path of the mobile robot, only increase in distance of sensor’s range by an amount which is sufficient for the mobile robot to ‘go through’ considering its dimensions has been considered as a ‘sudden point’ for finding the angle deviation.

Several changes in the algorithm (and also in the program) have been made for incorporating these modification, and used will be described in chapter-4.

CHAPTER-3

3.0 SYSTEM HARDWARE AND SOFTWARE OF ACTIVITYBOT AND LASER RANGE SENSOR

3.1 SYSTEM COMPONENTS USED IN THE PROJECT

The system hardware and software used in the present project are:

1. Parallax Activity-Bot mobile robot kit installed in the Robotics laboratory of Production Engineering department of Jadavpur University as shown in fig-3.1.
2. LASER range sensor (LIDAR lite V3) for providing the distance of an obstacle from the sensor.
3. Simple IDE software (**Version 1-1-0**) & Propeller C language to run the mobile robot

3.2 SYSTEM HARDWARE

This compact, zippy robot matches a multi-core Propeller microcontroller brain with great hardware. This robot's microcontroller contains free C language programming with SimpleIDE software for Windows and Mac. ActivityBot provides feedback of 360° High-speed servo motors with built-in encoders make fast, consistent maneuvers. In this robot we can Plug common electronic parts right into the breadboard or 3-pin headers and for that no soldering or special connectors needed. For navigation purpose the sensor is used with it is LASER range sensor (LIDAR lite V3). It's hardware contain built-in SD card slot and included microSD card are ready for data-logging and file storage.

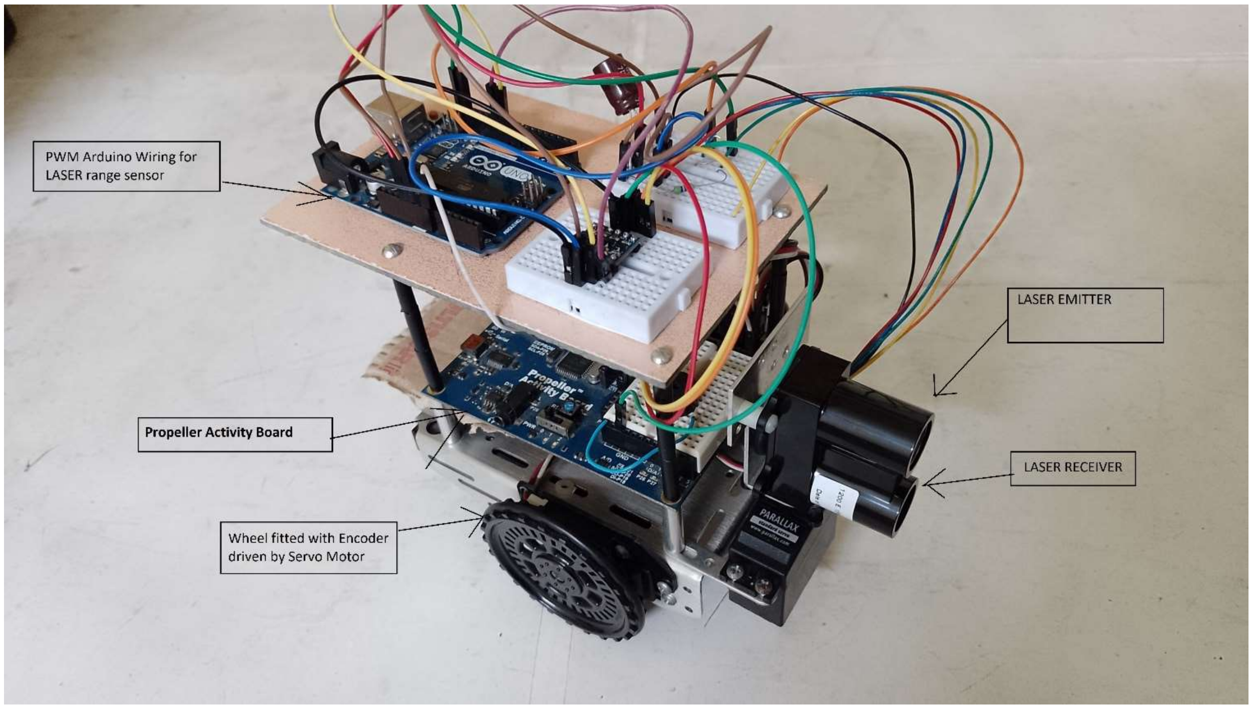

PWM Arduino Wiring for LASER range sensor
LASER EMITTER
Propeller Activity Board
LASER RECEIVER
Wheel fitted with Encoder driven by Servo Motor

Fig-3.1: Parallax Activity-Bot mobile robot kit with LASER range sensor.

The detailed component list is given below.

- Propeller Activity Board
- High speed servo
- ActivityBot encoder wheel with O-Ring tire
- LASER range sensor (LIDAR lite V3)
- Robot chassis
- Battery holder
- 1" tail wheel ball
- USB A to mini-B cable
- MicroSD card

3.2.1 Propeller Activity Board

Features:

- Built-in 8-core Propeller P8X32A microcontroller, 64 KB EEPROM, and 5 MHz crystal oscillator[30].
- Solder-free prototyping with breadboard and header sockets for power and I/O
- Six servo/sensor ports with power-select jumpers
- Automatically selects between USB and external power sources and provides USB over-current protection
- 6–15 V center-positive 2.1 mm barrel jack for external power supplies
- Convenient reset button and 3-position power switch
- Onboard mini stereo-audio jack with microphone/video pass-through
- Built-in micro SD card slot for data logging or storing WAV files
- Wireless Module socket accepts RF modules to simplify remote management
- Dedicated analog header sockets provide four A/D 12-bit inputs and two buffered variable resolutions D/A outputs
- Indicator lights show the status of system power, servo power, programming source, DAC output levels, wireless communication activity, and USB communication activity
- 3.3 V and 5 V switching voltage regulators with independent 1.8 amp outputs

Key Specifications:

- Power requirements: 6 to 15 VDC from an external power supply, or 5 V from a USB port
- Communication: USB mini-B (onboard serial over USB)
- Dimensions: 4.0 x 3.05 x 0.625 in (10.16 x 7.75 x 1.59 cm)
- Operating temp range: +32 to +158 °F (0 to +70 °C)

BOARD FEATURES

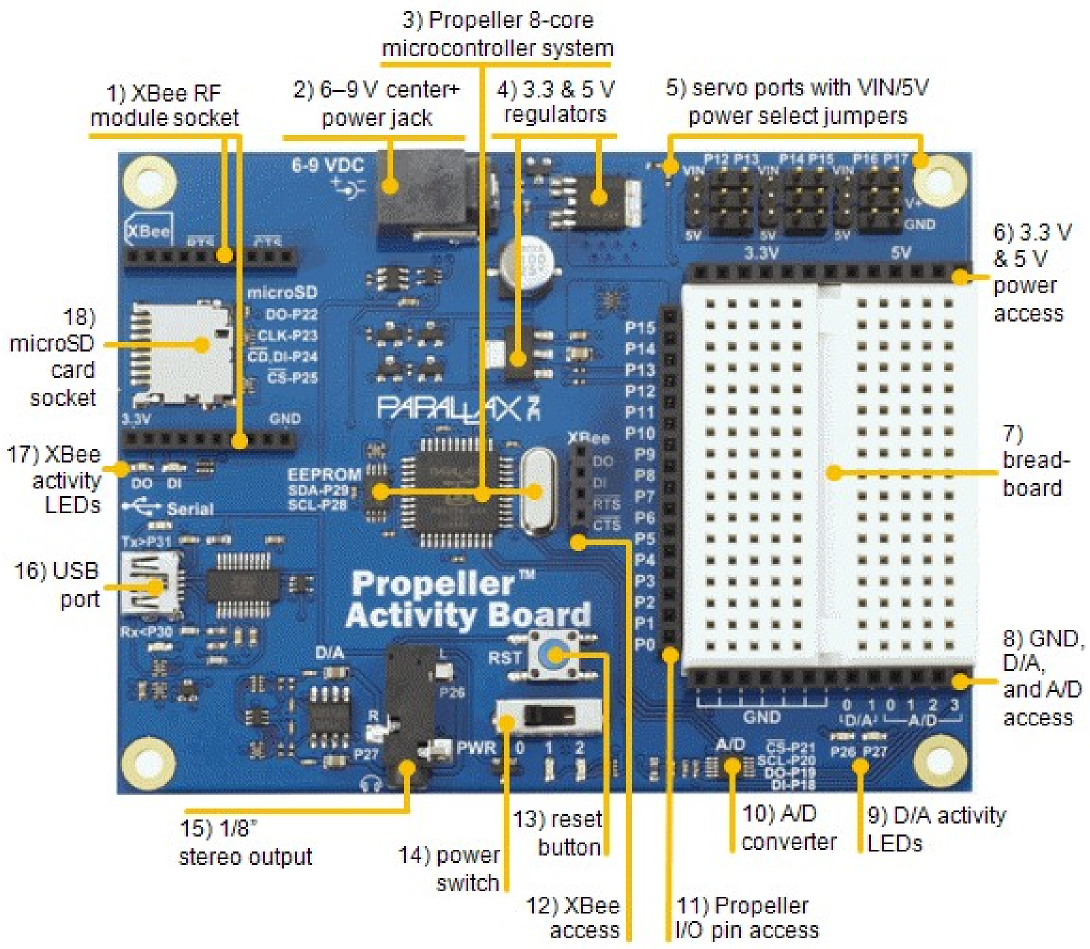


Fig-3.2: Propeller Activity Board.

I. Power Jack

The 2.1 mm centre-positive power jack is one of the two power input options. The board accepts 6–9 V from this connector. This option is useful for robots and other remote applications where the board is not powered from a computer’s USB port. Parallax’s 7.5 V, 1 A supply (#750-00009) works well with this board.

II. Propeller 8-core Microcontroller System

- 64 Kilobyte I2C EEPROM for program and data storage
- 8 core Propeller P8X32A microcontroller
- 5 MHz crystal oscillator

The P8X32A microcontroller has 8 cores, so it can do many different things at the same time. It uses I/O pins P28 and P29 to communicate with the I2C EEPROM for program and data storage. The crystal oscillator connected to the Propeller provides a clock signal for the system. The Propeller can multiply its 5 MHz oscillator signal by up to 16 for a system clock frequency of 80 MHz.

III. 3.3V & 5 V Regulators

The linear 5 V regulator can deliver up to 1.5 A with a 6 V power supply, or 750 mA with a 9 V power supply, for circuits built on the breadboard and devices connected to the servo ports. The 3.3 V regulator can deliver up to 500 mA for breadboard circuits, and it also powers the Propeller microcontroller system.

IV. Servo Ports

These ports are for connecting servos and other 3-pin devices to Propeller I/O pins, labelled above each port. Labels indicating the GND (ground) and PWR (power) pins for each port are along the right. Each pair of servo ports has a jumper on power-select pins to its immediate left. Each pair can be set to 5 V by placing the jumper over the pair of pins closer to the 5V label, or to unregulated input voltage from an external power input by placing it over the pair of pins closer to the VIN label. If the jumper for a pair of ports set to 5 V, they will receive regulated 5 V power whenever the power switch is set to 2. If the jumper for a pair of servo ports is set to VIN, they will receive unregulated power from the source connected to the 2.1 mm barrel jack, so long as the power switch is set to 2.

V. 3.3 V & 5 V Power Access

The positive 3.3 V and 5 V supply sockets are positioned along the top of the breadboard. Use jumper wires to connect these sockets to circuits you build on the breadboard.

VI. Breadboard

This breadboard has 34 5-socket rows arranged in 2 columns. The columns are separated by a valley in the middle. Any two wires plugged into the same 5-socket row become electrically connected. The socket spacing is 0.1”.

VII. GND, D/A, and A/D Access

- GND access sockets — use jumper wires to connect these sockets to circuits on the breadboard.
- Digital to Analog access sockets — D/A 0, 1
  - Output voltage range: 0 to 3.3 V.
  - D/A 0 is the digital to analog voltage from P26 after it has passed through a low-pass filter and buffer amplifier (but before it has passed through the coupling capacitor to the stereo output jack’s right speaker channel).
  - D/A 1 is the same as D/A 0, but the duty modulated signal is provided by P27.
- Analog to Digital access sockets — A/D 0, 1, 1, 2, 3
- Input voltage range: 0 to 5 V.

VIII. D/A Activity Lights

These yellow LEDs give a visual indicator of the output voltage at D/A sockets 0 and 1. They also indicate activity on the stereo output jack. The LEDs will vary in brightness with duty modulated digital to analog signals.

IX. Analog to Digital Converter

Use the Analog to Digital Converter to monitor the voltage at analog inputs labeled A/D 0, 1, 2, and 3. It will give a number from 0 to 4095, which tells what the voltage is in a range from 0 to 5 volts. The converter used here is a 12-bit, 200 ksps SPI ADC, with a 5 V reference.

X. Propeller I/O Pin

Access to Propeller I/O pins P0..P15. Use jumper wires to connect these I/O pins to circuits on the breadboard, or to the XBee access header.

XI. <u>XBee Access</u>

The XBee access header is to the left of the Propeller I/O pin access header. Use jumper wires between the two headers to connect Propeller I/O pins to XBee DO (data out), DI (data in), RTS (ready to send) and CTS (clear to send) pins.

XII. <u>Reset Button</u>

Use this button to restart the Propeller microcontroller's program. Press and hold to keep the microcontroller in reset, press and release to reset and allow the Propeller to load the program in EEPROM.

XIII. <u>Power Switch The power switch has 3 settings:</u>

- 0 — off
- 1 — power to the microcontroller system, including the P0-P15 via the I/O pin access socket
- 2 — power to the microcontroller system and servo ports; see 5) Servo Ports for details.
-

XIV. <u>USB Port</u>

The USB port is used:

- to load programs from your computer into the Propeller microcontroller
- to provide serial-over-USB communication with a terminal program on your computer.
- to supply 5 V power to the Propeller Activity Board from your computer's USB port. For power, the USB Port is input current limited to between 450 mA and 500 mA. This prevents any unexpected responses from USB 2.0 ports to current draws from motors, wiring mistakes, etc. If you are using this board with an external USB hub, be sure to use a powered hub if you are not providing power from the power jack.

XV. <u>XBee DO/DI Activity Lights</u>

These LEDs give a visual indicator of communication happening between the XBee module and the Propeller microcontroller. The XBee DO line activity is indicated with a blue LED. The XBee DI line activity is indicated with a red LED.

Propeller I/O Pin Assignments

I/0 Pin	**Function**
P0–P15	General-purpose I/O access alongside the breadboard
P12–P17	3-pin header signal pins — this is the servo port header above the breadboard
P18–P21	Analog to digital converter
P22	microSD card DO (data out)
P23	microSD card CLK (clock)
P24	microSD card DI (data in)
P25	microSD card /CS (active-low chip select)
P26–P27	P26–P27: Duty modulated D/A converter signals go to: – Logic buffered yellow LED circuits sockets for brightness control – Low-pass filter + op amp buffer with outputs ranging from 0 to 3.3 V: • To DA0 and DA1 analog outputs on J1 • Through coupling capacitor to stereo outputs
P28–P29	64 KB I2C EEPROM for program and data storage. P28 = CLOCK, P29 = DATA
P30	Propeller programming Rx (transmits signal received by USB-to-serial converter’s Rx line)
P31	Propeller programming/debugging Tx (receives signal transmitted by USB-to-serial converter’s Tx line)

3.2.2 High Speed Continuous Rotation Servo

The high speed servo motor is shown in the figure 3.3.

Features

- Bi-directional continuous rotation
- Up to 150 RPM @ 6 VDC, or 180 RPM @ 7.4 VDC
- Linear response to pulse-width modification for easy ramping
- 3-pin ground-power-signal cable and female header with 0.1" spacing for quick connection
- Easy to interface with any Parallax microcontroller
- Very easy to control; examples available for many programming languages

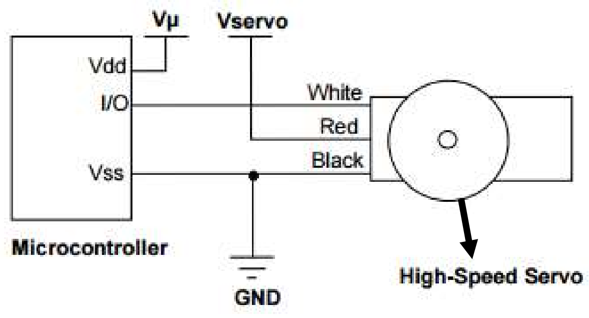
Key Specifications

- Power requirements:6.0 to 8.0 VDC;
  Maximum current draw
  130 +/- 50 mA @ 7.4 VDC
  when operating in no
  load conditions, 15 mA @ 7.4 VDC
  when in static state
- Communication: pulse-width modulation
- Speed: 0.30 +/- 0.06 sec/360°
- Torque: 22 +/-11 oz-in (1.6 +/- 0.8 kg-cm) @ 7.4 V
- Weight: 1.5 oz (42 g)
- Dimensions: approx- 2.2 x 0.8 x 1.6
  in (56 x 19 x 41 mm) excluding servo horn
- Operating temperature range: 14 to 113 °F (-10 to +45 °C)

fig-3.3: High Speed Servo Motor

Quick-Start Circuit:

Connection of the servo to the microcontroller as shown in the figure-3.4.



Vμ = microcontroller voltage supply

Vservo = 6 to 7.5 VDC, regulated or battery

I/O = PWM TTL or CMOS output signal, 3.3 to 5 V; <Vservo + 0.2 V

Fig-3.4: Circuit Diagram between servo with microcontroller

Servo Control

The Parallax Continuous Rotation Servo is controlled through pulse width modulation. Rotational speed and direction are determined by the duration of a high pulse, in the 1.3– -1.7 ms range. In order for smooth rotation, the servo needs a 20 ms pause between pulses. Fig-3.5 below is a sample timing diagram for a centered servo at 1.5 ms control pulse makes the servo stand still.

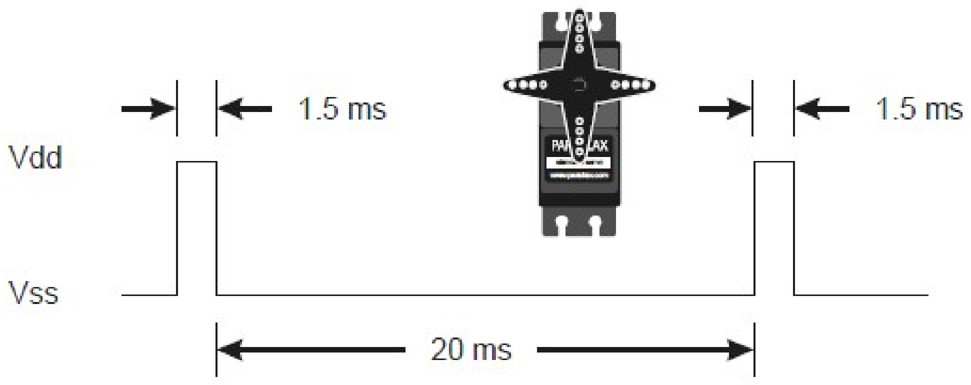


Fig-3.5

As the length of the pulse decreases from 1.5 ms, the servo will gradually rotate faster in the clockwise direction, as can be seen in the figure below: 3.6.

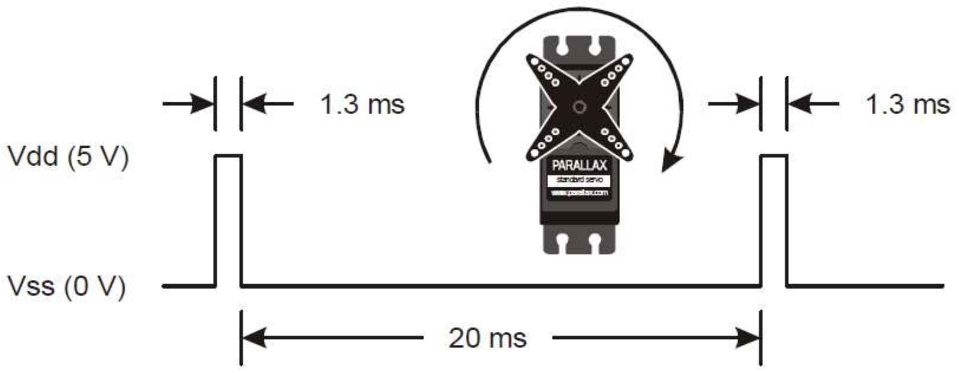


Fig-3.6.

Likewise, as the length of the pulse increases from 1.5 ms, the servo will gradually rotate faster in the counter-clockwise direction, as can be seen in the figure below: 3.7

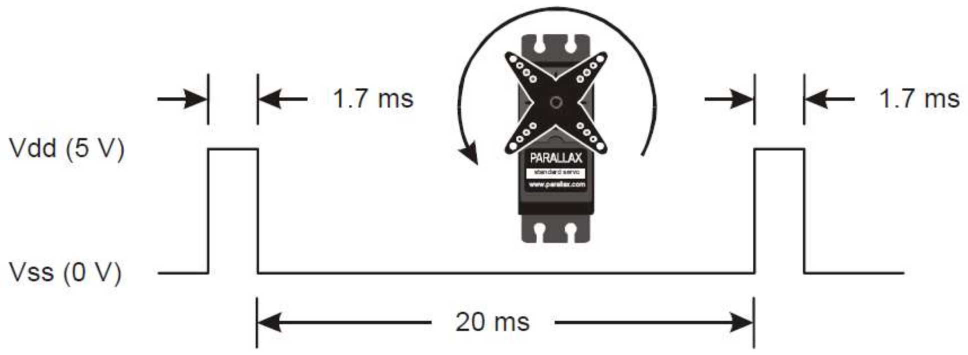


Fig-3.7

Relation between Velocity vs. Pulse Width :

The graph (fig-3.8) below shows that while the velocity response curve is consistent for each servo, the positive velocity response near 1.4 ms has a much sharper curve than the negative velocity response near 1.6 ms.

The net effect is that when two servos are mounted on a robot, pulse widths equidistant from the centre value of 1.52 ms will not result in the same rotational speed, and the robot will switch slightly to one side.

For robotics navigation applications that do not incorporate frequent course adjustment from object sensors, this can be compensated for in software, or with the use of encoders.

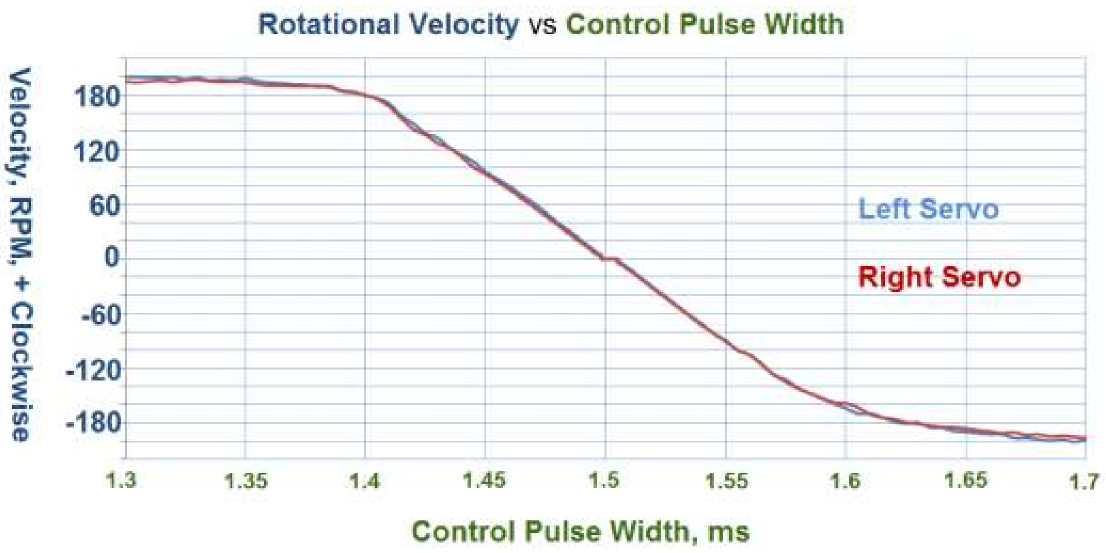


Fig-3.8: Graph between rotational velocity vs control pulse width for both servo

3.2.3 Encoder

Encoder Ticks

Each ActivityBot encoder shines infrared light at the ring of 32 spokes in the wheel next to it. If the light passes between the spokes, the encoder sends the Propeller a high signal. If it bounces off a spoke and reflects back to the encoder's light sensor, it sends a low signal to the Propeller. Each time the signal changes from high to low, or low to high, the Propeller chip counts it as an encoder tick.

Each encoder tick makes the wheel travel 3.25 mm forward. Remember, an encoder tick is counted when the encoder sensor detects a transition from spoke to hole or hole to spoke. Since there are 32 spokes and 32 holes, there are a total of 64 encoder ticks per wheel turn. Fig-3.9 shows a Cross section view of encoder wheel.

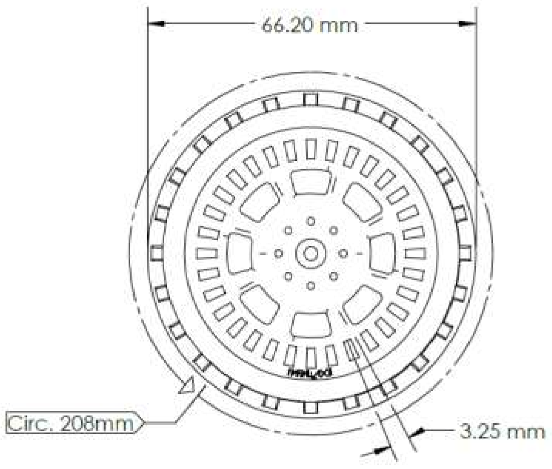


Fig-3.9: Cross Section View of Encoder Wheel

Sensing Direction

The Propeller chip knows what direction the servos turn based on the signal it uses to make the servo move. All it needs from the encoder is to know how fast it's turning. It does this by counting encoder ticks over a period of time. The libraries keep track of all this for you, so your programs just need to tell the robot how far or how fast to go.

3.2.4 LASER Range Sensor System:

LASER Range sensor system, also called LIDAR range sensor, which means Light Detection and Ranging, shown in the figure-3.10, provides an easy method of distance measurement. This sensor is perfect for any number of applications that require to perform distance measurements between the sensor and a stationary or moving objects.

Fig-3.10: LIDAR Range Sensor

Working Principle

This device measures distance by calculating the time delay between the transmission of a near-infrared laser signal and its reception after being reflected from a target using the known speed of light. It's unique signal processing approach transmits a coded signature and looks for that signature in the return, which allows for highly effective detection with eye-safe laser power levels. This sensor can measure up to Range 40m , with resolution ± 1cm and accuracy is ±2.5cm(for <5m) , ±10cm (for ≥ 5 m).

To take a measurement, this device first performs a receiver bias correction routine, correcting for changing ambient light levels and allowing maximum sensitivity. Then the device sends a reference signal directly from the transmitter to the receiver. It stores the transmit signature, sets the time delay for "zero"distance, and recalculates this delay periodically after several measurements. Next, the device initiates a measurement by performing a series of acquisitions. Each acquisition is a transmission of the main laser signal while recording the return signal at the receiver. If there is a signal match, the result is stored in memory as a correlation record. The next acquisition is summed with the previous result. When an object at a certain distance reflects the laser signal back to the device, these repeated acquisitions cause a peak to emerge, out of the noise, at the corresponding distance location in the correlation record. The device integrates acquisitions until the signal peak in the correlation record reaches a maximum value. If the returned signal is not strong enough for this to occur, the device stops at a predetermined maximum acquisition count. Signal strength is calculated from the magnitude of the signal record peak and a valid signal threshold is calculated from the noise floor. If the peak is above this threshold the measurement is considered valid and the device will calculate the distance, otherwise it will report 1 cm. When beginning the next measurement, the device clears the signal record and starts the sequence again.

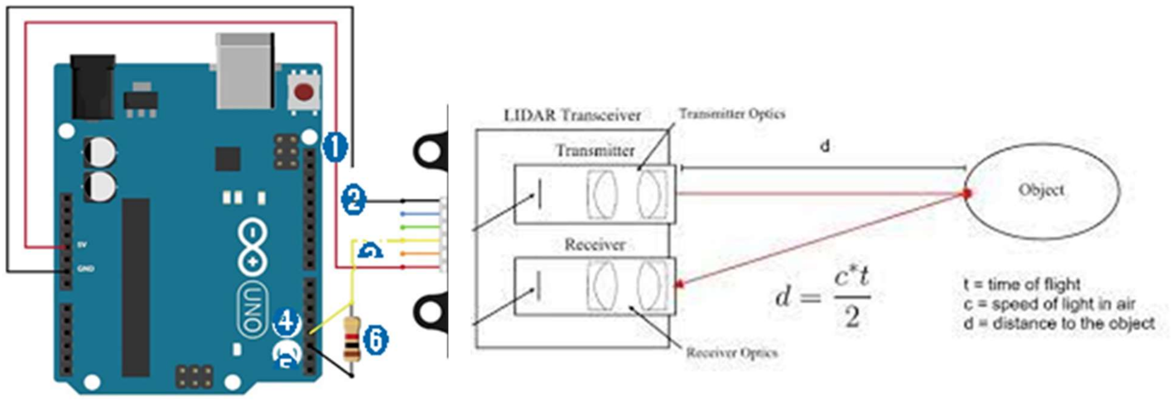


Fig-3.11: Working principle of LIDAR range sensor with PWM Arduino interfacing

Key Features:

- Range: 0-40m Laser Emitter
- Resolution: +/- 1 cm (0.4 in.)
- Accuracy: ±2.5cm(for <5m) , ±10cm (for ≥ 5 m)
- Power: 5 Vdc nominal 4.5 Vdc min., 5.5 Vdc max.
- Current Consumption: 105mA idle; 130mA continuous
- Rep Rate: 50 Hz default 500 Hz max
- Laser Wave Length/Peak Power: 905nm/1.3 watts
- Beam Divergence: 4m Radian x 2m Radian
- Optical Aperture: 12 × 2 mm (0.47 × 0.08 in.)
- Interface: I2C or PWM

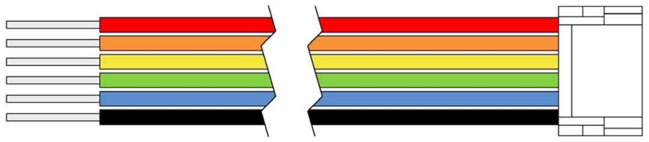
Physical Specifications

- Operating temperature: -20 to 60°C (-4 to 140°F)0 – 70° C.
- Size (LxWxH) : 20 × 48 × 40 mm (0.8 × 1.9 × 1.6 in.)
- Weight: 22 g (0.78 oz.)

LASER Specifications

- Wavelength: 905 nm (nominal)
- Total laser power (peak): 1.3 W
- Mode of operation: Pulsed (256 pulse max. pulse train)
- Pulse width: 0.5 µs (50% duty Cycle)
- Pulse train repetition frequency: 10-20 KHz nominal
- Energy per pulse: <280 nJ
- Beam diameter at laser aperture: 12 × 2 mm (0.47 × 0.08 in.)
- Divergence: 8 m Radian

Connections



Wire Color	Function
Red	5 Vdc (+)
Orange	Power enable (internal pull-up)
Yellow	Mode control
Green	I2C SCL
Blue	I2C SDA
Black	Ground (-)

3.3 ACTIVITYBOT SYSTEM SOFTWARE

The system software used to run the PROPELLER microcontroller is called Simple IDE software and the programming language used for controlling the ActivityBot this is called Propeller C.

3.3.1 Simple IDE

A USA based company Parallax Inc. has developed this Simple IDE software to control the ActivityBot (name of the mobile robot). It is available for Windows and Mac and even Linux. It's an open-source C programming environment for the multi-core Propeller microcontroller. This software helps in writing the simple C program in Propeller C and downloads the program into the Propeller microcontroller. The program can be downloaded through USB cable and then the loaded program is used to run the mobile robot either by loading the program into RAM or EEPROM. The program can be saved with a project file name and can be edited anytime as and when needed. The Simple IDE terminal shows the messages on screen for print source code and inputs can be entered for C input code.

3.3.2 Propeller C

This is a c-language version developed for 8-core propeller micro controller. There is some Header and Source files in this Propeller C which collects different functions and codes which are used in programming and fulfill the different tasks performed by the Simple IDE and Activity Bot.

3.3.3 Description of main Commands used in the Project

i) drive_setMaxSpeed (int speed)

Modifies the default maximum top speed for use with encoders. The default is 128 ticks/second = 2 revolutions per second (RPS). This is the full speed that drive_distance and drive_goto use. This value can currently be reduced, but not increased. Speeds faster than 128 ticks per second are "open loop" meaning the control system does not use the encoders to correct distance/speed.

Parameter :

Speed	Maximum cruising speed for drive_distance and drive_goto

ii) servo_angle (int pin, int degree Tenths)

For Parallax Standard Servo rotation angle can be set from 0 to 180 in tenths of a degree.

Examples:

servo_angle(pin, θ); // for θ degrees

servo_angle(pin, 900); // for 90 degrees

servo_angle(pin, 1800); // for 180 degrees

Parameters:

pin	Number of the I/O pin connected toservo.
degreeTenths (θ)	Tenths of a degree from 0 to 1800.

iii) drive_goto (int distleft, int distright)

This command make each wheel go a particular distance. Recommended for straight forward, backward, turns, pivots, and curves/arcs. This function ramps up to full speed if the distance is long enough. It holds that speed until it needs to ramp down. After ramping down it applies compensation. By default, this function does not return until the manoeuvre has completed.

Parameters:

distLeft	Left wheel distance in ticks (spoke to space and space to spoke transitions). Each "tick" transition is 1/64th of a wheel revolution, causing the wheel to roll approximately 3.25 mm
distRight	Right wheel distance in ticks

iv) Laser_cm

This commands gives measured distance of the nearest obstacle sensed by the LASER range sensor in centimetre.

Example: laser_cm()

v) pause (int time)

Delay cog from moving on to the next statement for a certain length of time. The default time increment is 1 ms, so pause(100) would delay for 100 ms = 1/10th of a second. This time increment can be changed with a call to the set_pause_dt function.

Parameters :

time	The number of time increments to delay.

.

3.4 ACTIVITYBOT NAVIGATION SYSTEM

In this section we will see how ActivityBot move forward direction, backward direction, and can rotate in left or right side. Before we have seen that the Parallax High Speed Continuous Rotation Servo is controlled through pulse width which decides the direction of rotation of servo motor. It also covers the LIDAR range sensor rotation attached at the front side of mobile robot which is used to sense the obstacle position during navigation. For movement purpose, first change the distance in terms of “ticks” for each wheel rotation. Positive ticks number for clockwise direction and negative ticks for counter clockwise direction.

The term “tick” indicates a transition from either spoke detected to hole detected. Activity Bot has 32 spokes, separated by 32 spaces, total 64 tics. If wheel rotates 1/64 th turn it will move 1tick means 3.25mm [fig-..]

Our programme needs the ticks value to tell the Activity Bot to move. Any distance can be converted into ticks value. ticks = distance mm ÷ 3.25 mm.

3.4.1 Forward and Backward movement of ActivityBot

For forward and backward movement of the activity Bot, the left and right both wheel will rotate in clockwise and anti-clockwise direction respectively. The abdrive library has a function named drive_goto. It is used to tell the Activity-Bot how far each wheel should turn in terms of 3.25 mm

increments. The details of straight line navigation of Activity-bot in forward and backward direction are described in figure-3.12

For forward movement

drive_goto(int distleft, int distright)   // distance value in ticks and both value is +ve.

For Backward movement

drive_goto(-int distleft,- int distright)   // distance value in ticks and both value is –ve.

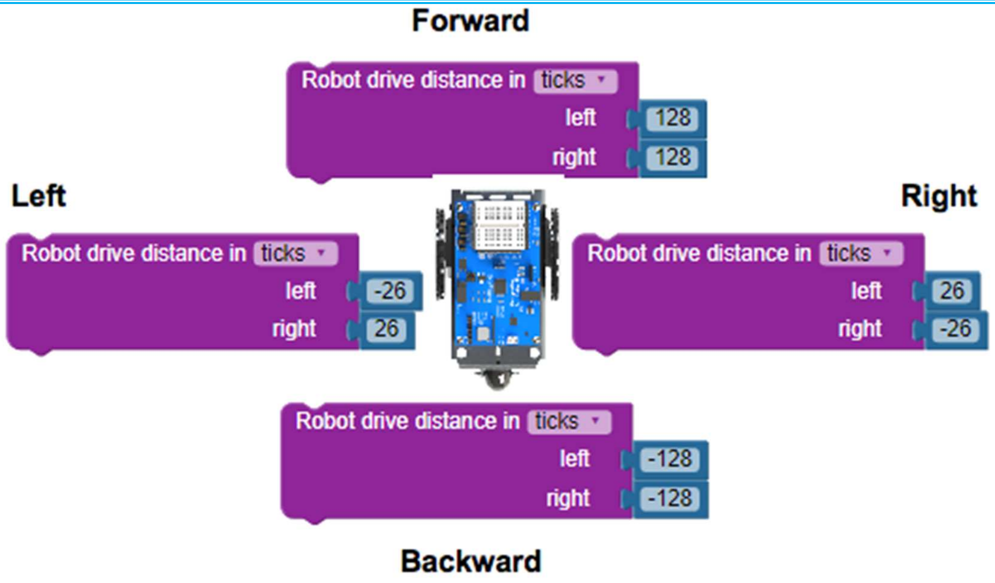

Forward
Robot drive distance in ticks
left 128
right 128
Left
Robot drive distance in ticks
left -26
right 26
Right
Robot drive distance in ticks
left 26
right -26
Robot drive distance in ticks
left -128
right -128
Backward

Fig-3.12 : Drive distance block to make the ActivityBot move.

3.4.2 Turning the ActivityBot

We can turn the activity Bot in two ways. One, robot can rotate in clockwise and anti-clockwise by holding one wheel still which is shown in the fig-3.13, and making other wheel turn about holding wheel. Two, by rotating the both wheel is opposite direction at same speed or at same ticks value which is shown in the figure-3.16.

Say for right wheel turn, the left wheel is to be held fixed, the right wheel will have to turn by 2*π*turning radius(R) which is equal to 664.76 mm where R= 105.8 mm. For turning activity-bot by n number of turn that would be 664.76 * n. To calculate the number of ticks for corresponding turning angle, no of ticks (t) = 664.76 * n / 3.25.

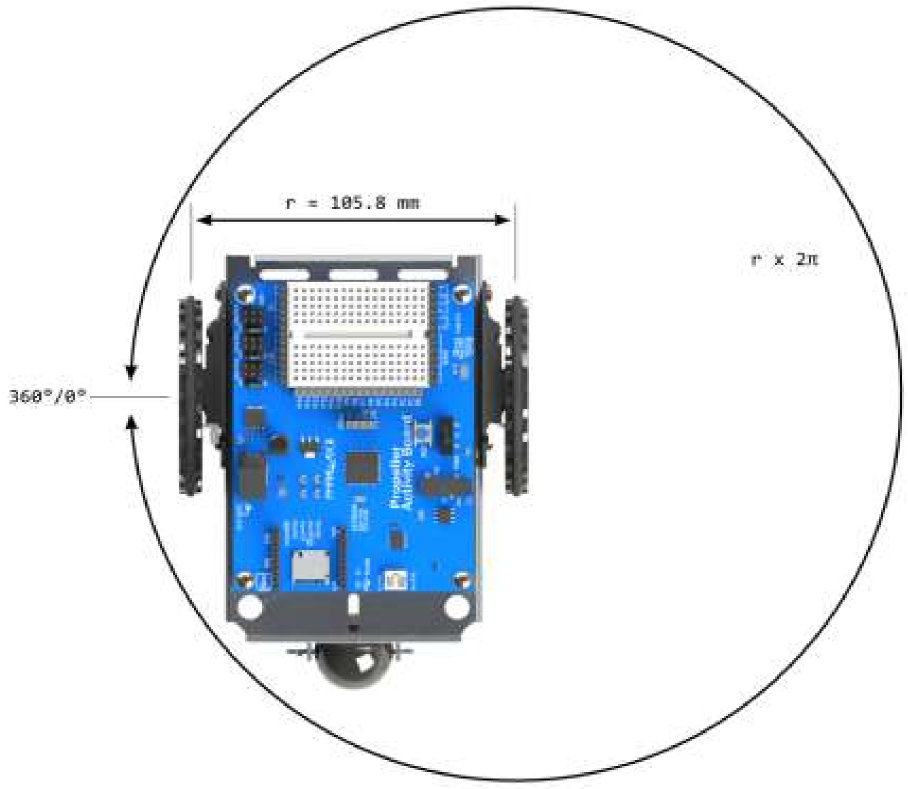


Fig-3.13: Rotation (clockwise) of mobile robot by holding right wheel still.

The command used for this turning is

drive_goto (left wheel ticks (t), 0) // for left turning (cw rotation)

drive_goto (0, right wheel ticks (t)) // for right turning (ccw rotation).

For rotation about centre of mobile robot, ticks will be divided into equal number of ticks for both wheel movement, but each wheel to be turned in opposite direction in respect of right or left turning.

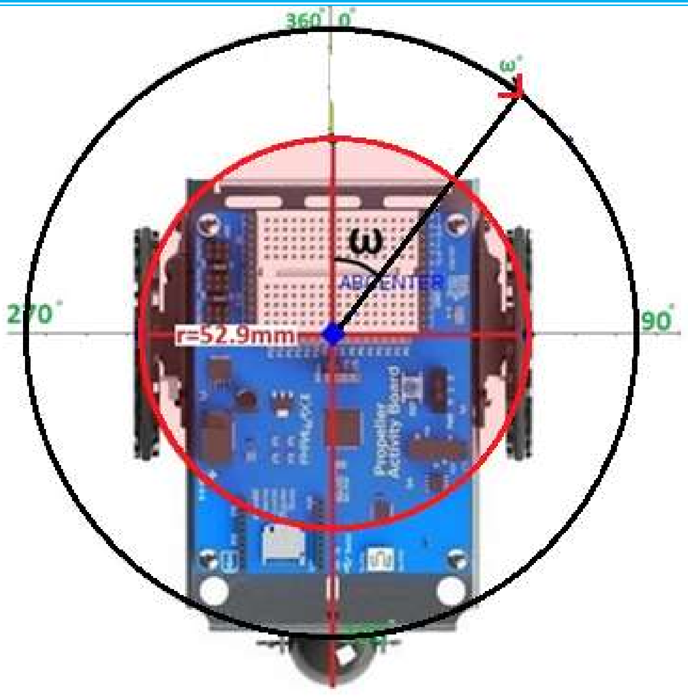


Fig- 3.14: Rotation about centre of mobile robot

The command uses for turning about mobile robot centre is

drive_goto (- t/2, t/2) // for right turning or ccw rotation.

drive_goto (t/2, - t/2) // for left turning or cw rotation.

CHAPTER-4

4.0 EXPERIMENTATION WITH PARALLAX ACTIVITYBOT AND LASER RANGE SENSOR FOR PATH PLANNING & NAVIGATION BASED ON MODIFIED POINTBUG ALGORITHM

4.1 MODIFICATION OF POINT BUG ALGORITHM FOR ACTUAL DIMENSION OF ROBOT.

In Point Bug and all other algorithm of Bug family the mobile robot is assumed as a point. But in real case when the algorithm is used for navigation of mobile robot avoiding obstacles and to reach the goal, one has to consider the real dimension of the mobile robot which is actually not a point . That's why some modifications have been done in point Bug algorithm considering the actual dimension of the mobile robot used in present object. When a sudden point is obtained after detecting an obstacle , the robot will have to move to a modified point which is slightly away from the actual sudden point.

4.1.1 Modification of Angle for Getting Modified Sudden Point

In the way to move towards goal if any obstacle is detected by range sensor, it starts scanning the obstacle to get sudden points. Now if it moves to the sudden point with minimum angle considering mobile robot as a point, then the robot will collide with obstacle for not considering actual robot dimensions. So a vital modification is carried out to avoid the collision with obstacle. An angle 'ang_{ext}' is added with both right and left side sudden point angle. The fig-4.1 is shown for right side angle only. LASER range sensor rotates about point 'Q' and robot rotates about point 'P'.

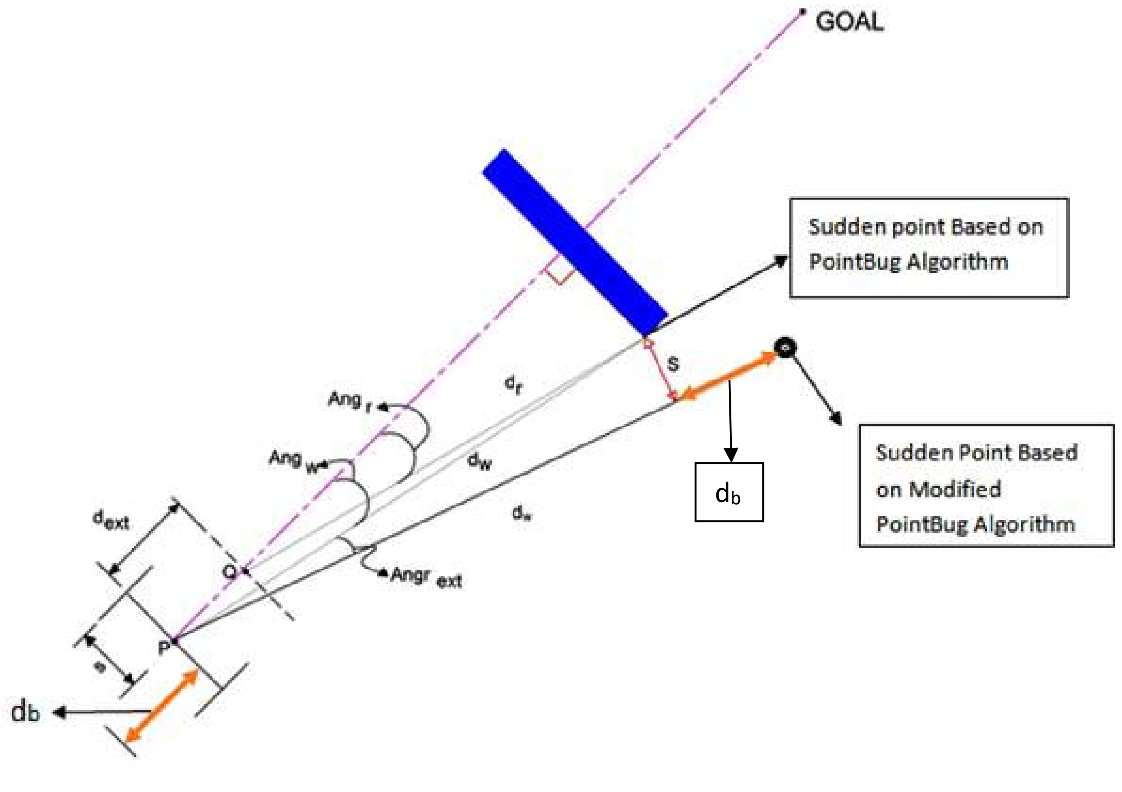

GOAL
Sudden point Based on PointBug Algorithm
S
d_r
Ang_r
Ang_w
d_w
d_w
d_b
Sudden Point Based on Modified PointBug Algorithm
d_{ext}
Q
$Angr_{ext}$
P
a
d_b

Fig-4.1: The geometry of angle and distance modification to avoid collision.

Q = Location of sensor at which point obstacle is detected.

P = Location of robot wheel axle centre at which point obstacle is detected.

Ang_r = Sudden point angle(right).

Ang_w = Angle of rotation required about wheel axle centre.

d_r = Sudden point distance from sensor.

d_w = Sudden point distance from wheel line center.

S = Distance between the wheel axle center (= 5 cm for activity Bot).

d_{ext} = Distance between sensor and wheel center along the line of movement of the robot (=7 cm).

$Angr_{ext}$ = Minimum addition angle rotation required to avoid collision considering robot dimension.

d_b = Distance between the wheel axle centre and rear end of the robot (= 8cm).

If we know the values of $d_{r,}$ ang_r and d_{ext} then we can easily find out the value of d_w and Ang_w from the figure-4.1

4.1.2 <u>Modification of Distance for Getting Modified Sudden Point</u>

After detecting sudden point, the mobile robot moves forward and reaches sudden point according to the LASER range sensor detected source point and in this situation if robot rotates towards goal and moves to the target, it will collide with obstacle. To avoid collision, distance modification is also important along with angle modification. So, after reaching sudden point with modified angle it will have to move some distance d_b(as shown in the fig-4.1) to reach at the modified sudden point for collision free movement from current position to target position.

4.2 PROGRAMMING DEVELOPED FOR PATH PLANNING OF MOBILE ROBOT BASED ON MODIFIED POINTBUG ALGORITHM

In this project path planning for ActivityBot mobile robot using LASER range sensor is developed using propeller-C language with the help of Simple IDE software. The Propeller-C program has been developed to guide the mobile robot from start point to goal point by considering modified PointBug algorithm in static environment where the obstacle position and size is unknown. Robot first orients itself at start point towards goal point direction and then starts moving towards goal step by step (one step = 3*3.25 mm). After each step, the distance of obstacle, if any, is measured by the range sensor. If any obstacle is detected by LASER range sensor within a specified range (taken as 30 cm), it will stop moving and the LASER range sensor will start searching for the sudden point by rotating itself both sides. In this present work sudden point is taken as a point from which the next obstacle distance measured by range sensor is more than a specific distance which enough for the mobile robot to go through. At the same time the angular rotation of the sudden point is being determined by recording the angular rotation of sensor during scanning. After scanning the first obstacle, it is rotated further in the same direction for searching the presence of any other obstacle within specified angle (taken as 30o at 30cm distance) such that there is enough gap for the robot to go through.

Within this range of angle, if any obstacle is detected then this process will be repeated and sudden point will be modified. If no sudden point is found within 90o rotation of range sensor then a message will be shown to reply that is not possible to move that direction. And this process is repeated for the both sides of the robot.

After detecting sudden point on both sides by, the direction that makes minimum angle to rotate based on PointBug algorithm will be selected and the modified sudden point is determined by considering actual dimension of robot based on modified point Bug algorithm which is discussed in the section 4.1.1 and 4.1.2.

After rotating to the direction of modified sudden point from the current direction the robot moves to that point and after reaching that point it rotates towards the goal point. Then again the robot starts moving towards the goal point step by step as it did from start point. The whole process is repeated until the robot reaches the goal point or a situation arrives when there is no path. The simplified flowchart of the process is shown in figure 4.2 and the computer program in propeller-C is given in section 4.4.

4.3 FLOWCHART OF DEVELOPED PROGRAM

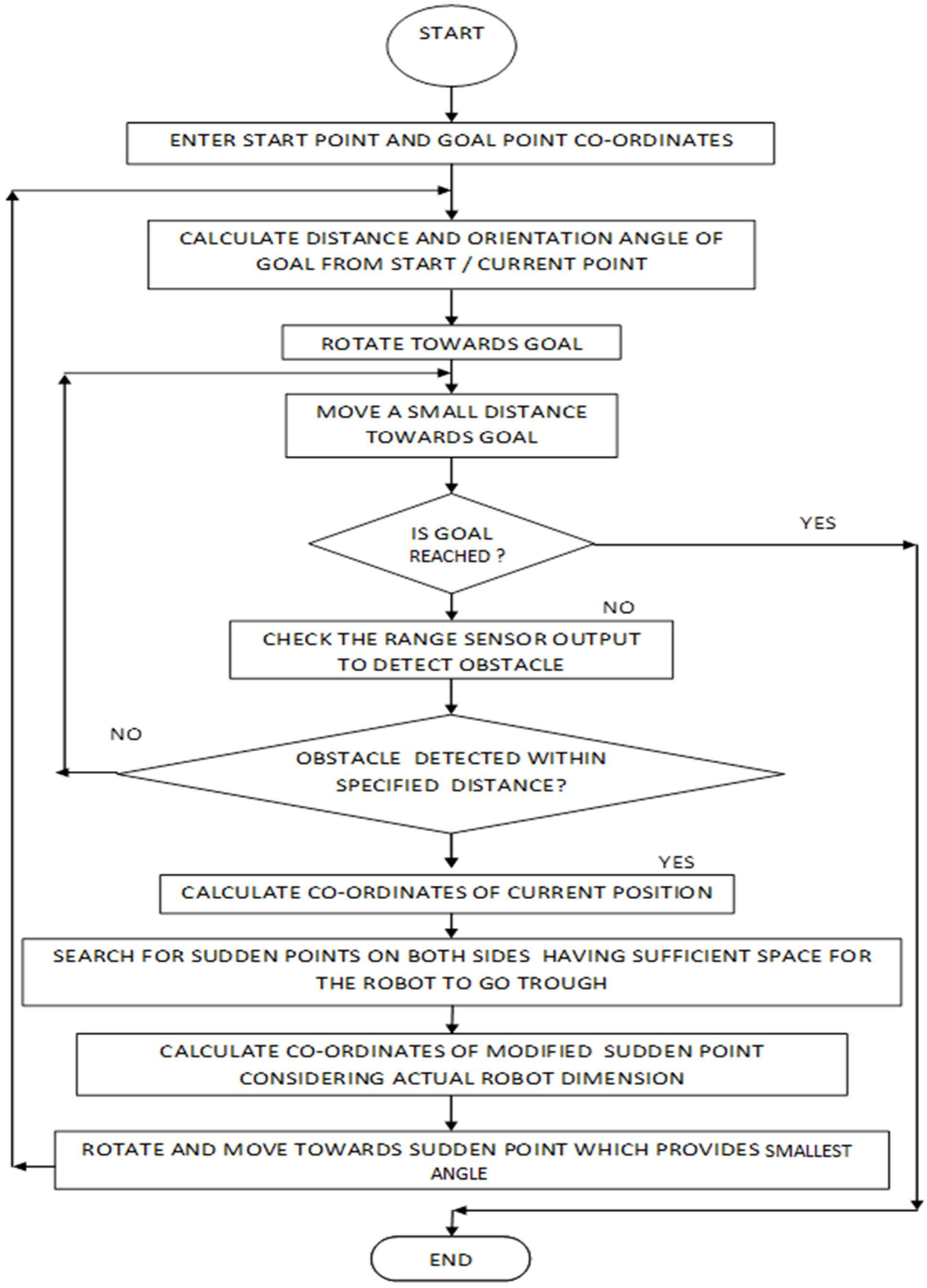

START
ENTER START POINT AND GOAL POINT CO-ORDINATES
CALCULATE DISTANCE AND ORIENTATION ANGLE OF GOAL FROM START / CURRENT POINT
ROTATE TOWARDS GOAL
MOVE A SMALL DISTANCE TOWARDS GOAL
IS GOAL REACHED ?
YES
NO
CHECK THE RANGE SENSOR OUTPUT TO DETECT OBSTACLE
NO
OBSTACLE DETECTED WITHIN SPECIFIED DISTANCE?
YES
CALCULATE CO-ORDINATES OF CURRENT POSITION
SEARCH FOR SUDDEN POINTS ON BOTH SIDES HAVING SUFFICIENT SPACE FOR THE ROBOT TO GO TROUGH
CALCULATE CO-ORDINATES OF MODIFIED SUDDEN POINT CONSIDERING ACTUAL ROBOT DIMENSION
ROTATE AND MOVE TOWARDS SUDDEN POINT WHICH PROVIDES SMALLEST ANGLE
END

Fig 4.2 Flowchart of Developed Program

<u>4.4 PROPELLER C PROGRAM FOR PATH PLANNING OF ACTIVITYBOT MOBILE ROBOT IN PRESENCE OF OBSTACLES USING LASER RANGE SENSOR BASED ON MODIFIED POINTBUG ALGORITHM</u>

The complete C program developed for the project is given below:

```
/*
PATH PLANNING OF ACTIVITY ROBOT USING A LASER RANGE SENSOR BASED ON   MODIFIED
POINT-BUG ALGORITHM
/*

#include "simpletools.h"   // Include simple tools
#include "abdrive.h"       // Include ActivityBot files
#include "servo.h"         // Include servo motor files
#include "fdserial.h"      // Include files for serial communication
 //                           required for LASER range sensor

int main()              // Main function
{
 float X1, X2, Y1, Y2,ang,angrad, n, tt ,delx,dely, d, pi=3.1415926;
 int nticks=3,t;
 drive_setMaxSpeed(4);

 print("Enter start point(x,y) in mm");
 scan("%f%f\n",&X1,&Y1);
 print("Enter goal point(x,y) in mm");
 scan("%f%f\n",&X2,&Y2);

 delx=X2-X1;
 dely=Y2-Y1;
 d=sqrt((X2-X1)*(X2-X1)+(Y2-Y1)*(Y2-Y1));
 print("total distance = %f mm \n",d);
 angrad=atan2(dely,delx);
 ang=angrad*180/pi;      // in degree
 print("angle=%f\n",ang);

 n=ang/360;
 tt=(664.76*n)/3.25;
 t=tt/2;
 int i,m;
 float dis, D;
 servo_angle(16,900);
 pause(1000);
 drive_goto(-t,t);
 pause(500);

 L1:
  i=0;
  m=d/(nticks*3.25);
  print("m=%d\n",m);
```

```
         if (i<m )
           {
                drive_goto(nticks,nticks);
                dis= 10*laser_cm();
                print("dis=%f\n",dis);
                i=i+1;
                 if (dis>300)
                      goto L1;
           }
         else
           {
                 goto L0;
           }
D=i*(nticks*3.25);
print("Distance Moved=%f\n",D);

X1=X1+D*cos(angrad);
Y1=Y1+D*sin(angrad);

d=sqrt((X2-X1)*(X2-X1)+(Y2-Y1)*(Y2-Y1));
print("d=%f\n",d);

float j,k,a,a0,b,b0,angr,angl,angt,dr,dl;
a=b=a0=b0=dis;

j=900;
L3:
   while((a-a0)<=200)
   {
     servo_angle(16,j);
     pause(500);
     a0=a;
     a=10*laser_cm();

     angr=(900-j)/10;
     print("%d  %d\n",angr,a);
     j=j-10;
     if (j<=0)
     {
      print("no sudden point is found on right side  \n");
      angr=100;
      goto L4;
     }
   }

  angr=angr-1;

  while ((a-a0)>=200)
  {
     servo_angle(16,j);
     pause(500);
     a=10*laser_cm();
     angt=(900-j)/10;
     print("%d  %d\n",angt,a);
     j=j-10;
     if ((angt-angr)>30)
     {
     print("angr=%d dist=%d\n", angr,a0);
     goto L4;
     }
  }
   goto L3;
```

```
 L4:
     pause(500);
     servo_angle(16,900);

         a=a0+120;
          angr=angr+15;
         print("angr=%f\n",angr);
         print("a=%f\n",a);

k=900;
L5:
 while((b-b0)<=200)
   {
     servo_angle(16,k);
     pause(500);
     b0=b;
     b=10*laser_cm();
     angl=(k-900)/10;
     print("%d  %d\n",angl,b);
     k=k+10;
     if (k>=1800)
     {
      print("no sudden point is found on left side  \n");
      angl=100;
      goto L6;
     }
   }
  angl=angl-1;

  while ((b-b0)>=200)
  {
     servo_angle(16,k);
     pause(500);
     b=10*laser_cm();
     angt=(k-900)/10;
     print("%d  %d\n",angt,b);
     k=k+10;
     if ((angt-angl)>30)
     {
     print("angl=%d dist=%d\n", angl,b0);
     goto L6;
     }
  }
  goto L5;

 L6:
     pause(500);
     servo_angle(16,900);

         b=b0+120;
          angl=angl+15;
         print("ang2=%f\n",angl);
         print("b=%f\n",b);
```

```
if(angr<angl)
           {
           int t1,tto,tt1,ta ;
           float n1,no,to,p1,angturned;
           n1= angr/360;
           t1= (664.76*n1)/3.25;
           tt1=t1/2;
           ta=a/3.25;

           drive_goto(tt1,-tt1);
           pause(100);
           drive_goto(ta,ta);
           pause(20);

           p1=ang-angr;
           X1=X1+(a*cos(p1*pi/180));
           Y1=Y1+(a*sin(p1*pi/180));
           print("%f\n %f\n",X1,Y1);
           print("p1=%f\n",p1);

           d=sqrt((X2-X1)*(X2-X1)+(Y2-Y1)*(Y2-Y1));
           delx=X2-X1;
           dely=Y2-Y1;
           angrad=atan2(dely,delx);
           ang=angrad*180/pi;
           angturned=ang-p1;

           print("d=%f\n ang=%f\n angtur=%f\n", d,ang,angturned);

           no=angturned/360;
           to=(664.76*no)/3.25;
           tto=to/2;

          drive_goto(-tto,tto);
          pause(100);

          if (d>0)
               goto L1;

           }
```

```
    else
            {
            int t2,tto,tt2,tb ;
            float n2,u,no,to,p2,angturned;

            n2= angl/360;
            t2= (664.76*n2)/3.25;
            tt2=t2/2;
            tb=b/3.25;

            drive_goto(-tt2,tt2);
            pause(100);
            drive_goto(tb,tb);
            pause(20);

            p2= ang+angl;
            X1=X1+(b*cos(p2*pi/180));
            Y1=Y1+(b*sin(p2*pi/180));
            print("%f\n %f\n",X1,Y1);
            print("p2=%f\n",p2);

            d=sqrt((X2-X1)*(X2-X1)+(Y2-Y1)*(Y2-Y1));
            delx=X2-X1;
            dely=Y2-Y1;
            angrad=atan2(dely,delx);
            ang=angrad*180/pi;
            angturned=p2-ang;

            print("d=%f\n ang=%f\n angtur=%f", d,ang,angturned);

            no=angturned/360;
            to=(664.76*no)/3.25;
            tto=to/2;

            drive_goto(tto,-tto);
            pause(100);

    if (d>0)
             goto L1;

            }

  L0:
  print ("goal reached \n");
  pause(1000);

}
```

Different variables used in the program:

X1, Y1: Co-ordinates of starting or current position (initially the start position).
X2, Y2: Co-ordinates of goal position.
ang: goal orientation angle with respect to current orientation in angle.
angrad: goal orientation angle with respect to current orientation in radian.
n: no of turns corresponding to angle (ang).
tt: no of encoder ticks for rotating “n” no of turn.
t: half value of ticks (tt) for both wheel rotation in opposite direction about centre of robot wheel axis.
delx, dely: incremental value of x and y co-ordinates from current and goal position.
d: Euclidean distance between current (initially the start position) and goal position.
nticks: no of encoder ticks for straight line movement.
i: counter variable to store the number of ticks travelled so far.
m: no of ticks for travelling d distance.
dis: obstacle distance obtained from range sensor.
D: distance moved till it detects obstacle.
j: counter variable for scanning sudden point angle on right.
k: counter variable for scanning sudden point angle on left.
a: right sudden point distance.
a0: previous value of right sudden point distance.
b: left sudden point distance
b0: previous value of left sudden point distance.
angr: right sudden point angle measured from normal direction.
angl: left sudden point angle measured from normal direction.
dr: summed distance of current to right sudden and from right sudden to target point.
dl: summed distance of current to left sudden and from left sudden to target point.
n1: no of turns corresponding to angle (ang1).
n2: no of turns corresponding to angle (ang2).
t1: no of encoder ticks for rotating “n1” no of turn.
t2: no of encoder ticks for rotating “n2” no of turn.
tt1: half value of ticks (t1) for both wheel rotation in opposite direction about centre of robot wheel axis.
tt2: half value of ticks (t2) for both wheel rotation in opposite direction about centre of robot wheel axis.
ta: no of ticks for travelling a distance.
tb: no of ticks for travelling b distance.
p1: angle of robot rotation after moving to right sudden point measured from X-axis.
p2: angle of robot rotation after moving to left sudden point measured from X-axis.
angturned: angle of rotation of robot from sudden point to the target point calculated from X-axis.
no: no of turn corresponding to angle (angturned) .
to: no of encoder ticks for rotating “no” no of turn.
tto: half value of ticks (to) for both wheel rotation in opposite direction about centre of robot wheel axis.

4.5 EXPERIMENT RESULTS AND DISCUSSIONS

The experiment and its related all modifications have been done in the Robotics lab of Production Engineering department, Jadavpur University. After developing the program, it has been run successfully and the ActivityBot is guided to follow a collision free path from start point to goal point based on modified pointBug algorithm in different layout. At the end we have achieved our aim.

The developed C program has been run for moving the ActivityBot robot from its starting point to the goal point by detecting any obstacle using LASER range sensor. The angular position of the sudden point is obtained from servo motor rotation on which the LASER sensor is mounted. Initially, the robot orientation is parallel to the x-axis and it rotates towards goal by turning through required angle, and calculating the numbers of ticks and using drive_goto command for rotating cw or ccw direction accordingly. The forward movement is executed using drive_goto command. While advancing towards goal the range sensor continuously senses the distance to detect any obstacle within a specified distance. On detection of obstacle, scanning operation is carried out by simply rotating the servo motor attached to the sensor for determining the sudden points on both sides. Sudden point angle and distance as retrieved from the sensor are modified, as described in sections 4.1.1 and 4.1.2 The robot then takes decision about its next point to move based on the calculation of lengths of sub-paths of current to target point through one modified sudden point and will move towards that sudden point following the selected path.
On reaching the sudden point, the program again calculates the angle and distance of goal from current point, then rotates and starts moving towards goal. The program will continue till the robot reaches its target. In this way the robot moves in an unknown environment by avoiding obstacle in near optimal path. The developed program has been run successfully for different layouts of the workspace with different positions of the obstacles and goal point.

CHAPTER-5

5.0 CONCLUSION AND FUTURE SCOPE

5.1 CONCLUSION

Based on the foregoing analysis, study of the algorithm, program development, experimentations and results on “LOCAL PATH PLANNING OF A MOBILE ROBOT USING LASER RANGE SENSOR BASED ON MODIFIED POINT BUG ALGORITHM” for the present project, the following conclusions may be drawn.

1. Various algorithms of different path planning techniques have been studied thoroughly including bug algorithms, where point bug has been found suitable in most cases and has the capability of robot navigation methods for local path planning with using minimum number of sensors.

2. Point bug algorithm has some limitations in finding optimal path because minimizing the angular deviation does not always guarantee shortest path. But in various cases, this algorithm may give the optimal path depending upon the layout of obstacles.

3. Modified PointBug algorithm is used in the present project. In most of the path planning algorithms the mobile robot is consider as a point but here in modified Point Bug algorithm robot actual dimensions are considered.

4. Because of considering robot dimension, modification of angle of rotation of robot along sudden point and modification of distance move along sudden point have been done to avoid collision.

5. An arrangement has been made for setting up the workspace consisting of the mobile robot (ActivityBot) and multiple obstacles for various layout of workspace.

6. A program in C, based on Modified PointBug algorithm for navigating the robot from start to goal position in presence of static obstacle has been developed using Simple IDE software.

7. The developed program has been run successfully for different workspace layout.

5.2 FUTURE SCOPE

The design, analysis, algorithm, techniques described in the present thesis is applicable to any mobile robot with two wheels driven by separate motors. However, the program and the experimental results described in the present thesis is applicable only for the specific mobile robot used in the present project. But with hardware based modification of the present program, any type of wheeled mobile robot may be navigated in the presence of obstacles. Further scope of the present project includes use vision system to solve the problems associated dynamic obstacles.

CHAPTER-6

6.0 REFERENCES

References :

[1] Navid Toufan, Aliakbar Niknafs, *“Robot path planning based on laser range finder and novel objective functions in grey wolf optimizer”,* SN Applied Sciences volume 2, Article number: 1324 (2020).

[2] Aisha Muhammad , Mohammed A. H. Ali ,* , Sherzod Turaev , Rawad Abdulghafor , Ibrahim Haruna Shanono , Zaid Alzaid , Abdulrahman Alruban , Rana Alabdan , Ashit Kumar Dutta and Sultan Almotairi , “*A Generalized Laser Simulator Algorithm for Mobile Robot Path Planning with Obstacle Avoidance*” Sensors 2022, 22, 8177. https://doi.org/10.3390/s22218177

[3] James Ng & Thomas Bräun, “*Performance Comparison of Bug Navigation Algorithms”*, Received: 22 January 2007 / Accepted: 11 March 2007 /Published online: 25 April 2007.

[4] Zohaib, M., Pasha, M., Riaz, R.A., Javaid, N., Ilahi, M., & Khan, R.D, *“Control Strategies for Mobile Robot With Obstacle Avoidance”* COMSATS Institute of Information Technology, Islamabad, Pakistan. *CoRR, abs/1306.1144*, 2013.

[5] Muhammad Zohaib, Syed Mustafa Pasha, Nadeem Javaid, Jamshed Iqbal, “*Intelligent Bug Algorithm (IBA): A Novel Strategy to Navigate Mobile Robots Autonomously*,” International Multi Topic Conference (IMTIC 2013), Communication Technologies, Information Security and Sustainable Development, Pages 291-299.

[6] Donghyun Lee, Jongmin Jeong, Yong Hwi Kim, Jin Bae Park, "*An improved artificial potential field method with a new point of attractive force for a mobile robot*", *Robotics and Automation Engineering (ICRAE) 2017 2nd International Conference on*, pp. 63-67, 2017

[7] Dib, Lynda, “*E-Bug: New Bug Path-planning algorithm for autonomous robot in unknown environment”* University of Badji-Mokhtar, Annaba, 2015; https://www.researchgate.net/publication/289335757

[8] Mohamad, Z, “*Point to Point Sensor Based Path Planning Algorithm for Autonomous Mobile Robots”* Fac. of Elec. Engineering, Fac. of Mech. Engineering, University Teknologi MARA, MALAYSIA, 2010.

[9] Leena.N , K.K.Saju , "*A survey on path planning techniques for autonomous mobile robots* ," IOSR Journal of Mechanical and Civil Engineering (IOSR-JMCE) e-ISSN: 22781684, p-ISSN: 2320-334X PP 76-79, www.iosrjournals.org.

[10] Jun-Hao Liang, Ching-Hung Lee, *"Efficient Collision-Free Path-Planning Of Multiple Mobile Robots System Using Efficient Artificial Bee Colony Algorithm*," Advances In Engineering Software 79 (2015), Pages 47–56.

[11] C. Balaguer, A. Martí, "*Collision-Free Path Planning Algorithm For Mobile Robot Which Moves Among Unknown Environment,"* Robot Control 1988 (Syroco '88), Selected Papers From The 2nd IFAC Symposium, Karlsruhe, FRG, 5–7 October 1988, A Volume In IFAC Symposia Series 1989, Pages 261–266.

[12] Farouk MEDDAH, Lynda DIB, "*P*: A New Path Planning Algorithm for Autonomous Robot in an Unknown Environment",* International Journal of Advances in Computer Science & Its Applications– IJCSIA Volume 5: Issue 1 [ISSN: 2250-3765], April 2015, Pages 15-18.

[13] Hoc Thai Nguyen, Hai Xuan Le, *" Path planning and Obstacle avoidance approaches for Mobile robot,"* IJCSI International Journal of Computer Science Issues, Volume 13, Issue 4, july 2016, ISSN (Print); 1694-0814, ISSN (Online): 1694-0784.

[14] Muhammad Zohaib, Syed Mustafa Pasha, Nadeem Javaid, Jamshed Iqbal, "*Intelligent Bug Algorithm (IBA): A Novel Strategy to Navigate Mobile Robots Autonomously,"* International Multi Topic Conference (IMTIC 2013), Communication Technologies, Information Security and Sustainable Development, Pages 291-299.

[15] Zi-Xing Cai, Zhi-Qiang Wen, Xiao-Bing Zou, Bai-Fan Chen, "*A Mobile Robot Path Planning Approach Under Unknown Environments*," Proceedings Of The 17th World Congress The International Federation Of Automatic Control, Seoul, Korea, July 6-11, 2008.

[16] Yang-Ge Wu, Jing-Yu Yang, Ke Liu, "*Obstacle Detection And Environment Modeling Based On Multi Sensor Fusion For Robot Navigation*," Artificial Intelligence In Engineering 10 (1996), Pages 323-333.

[17] James Ng, "*A Practical Comparison of Robot Path Planning Algorithms given only Local Information"*, available at: robotics.ee.uwa.edu.au/theses/2005-Navigation-Ng.pdf

[18] Jang Gyu Lee, Hakyoung Chung, "*Global Path Planning For Mobile Robot With GridType World Model*," Robotics And Computer-Integrated Manufacturing, Volume11, Issue1, March 1994, Pages 13-21.

[19] Takanori Shibata, Toshio Fukuda, Kazuhiro Kosuge, Fumihito Arai, "*Path-Planning For Multiple Mobile Robots By Using Genetic Algorithms, Robotics*," Mechatronics And Manufacturing Systems 1993, Pages 409–414.

[20] Zhao-Qing Ma, Zeng Ren Yuan, "*Real-Time Navigation And Obstacle Avoidance BasedOn Grids Method For Fast Mobile Robot*," Algorithms And Architectures For Real-Time Control 1992, Preprints Of The IFAC Workshop, Seoul, Korea, 31 August – 2 September 1992, A Volume In IFAC Postprint Volume 1992, Pages 205–210.

[21] B. Margaret Devi, Prabakar S, *"Dynamic Point Bug Algorithm For Robot Navigation*", International Journal of Scientific & Engineering Research, Volume 4, Issue 4, April-2013, Pages 1276-1279.

[22] Lei Cai, Juanjuan Yang, Li Zhao, Lan wu, "*An efficient optimization algorithm for quadratic programming problem and its applications to mobile robot path planning*", International Journal of Advanced Robotic Systems, January-February 2018, http://journals.sagepub.com/home/arx.

[23] Rami Al-Jarrah, Mohammad Al-Jarrah, and Hubert Roth, "*A Novel Edge Detection Algorithm for Mobile Robot Path Planning*", Hindawi Journal of Robotics Volume 2018, Article ID 1969834, 12 pages https://doi.org/10.1155/2018/1969834.

[24] Valencia R., Andrade-Cetto J. “*Path Planning in Belief Space with Pose SLAM. In: Mapping, Planning and Exploration with Pose SLAM*”, Springer Tracts in Advanced Robotics, volume 119, springer, cham, 2018.

[25] Li G, Chou W. *“Path planning for mobile robot using self-adaptive learning particle swarm optimization”*, Science China Information Sciences May 2018, 1;61(5):052204.

[26] Haj Darwish, Ahmed, Abdulkader Joukhadar, and Mariam Kashkash. *"Using the bees algorithm for wheeled mobile robot path planning in an indoor dynamic environment." Cogent Engineering* just accepted (2018): 1426539.

[27] Akka, Khaled, and Farid Khaber*. "Mobile robot path planning using an improved ant colony optimization." International Journal of Advanced Robotic Systems* 15.3 (May 2018): 1729881418774673.

[28] Roy, Nirmalya, et al. *"Implementation of Image Processing and Reinforcement Learning in Path Planning of Mobile Robots." International Journal of Engineering Science* 15211, Oct 2017.

[29] Kuisong Zheng , Feng Wu and Xiaoping Chen. " *Laser-Based People Detection and Obstacle Avoidance for a Hospital Transport Robot* " *Sensors 2021, 21(3), 961; https://doi.org/10.3390/s21030961*.

[30] Parallax Activity Bot reference manual and simple IDE software details and syntax of the commands and their functions are available at https://learn.parallax.com.