ONLINE PATH PLANNING OF A MOBILE ROBOT USING LASER RANGE SENSOR BASED ON AN IMPROVED E-BUG ALGORITHM

BY

SOURAV ADHIKARY
B. Tech in Mechanical Engineering, 2018
Meghnad Saha Institute of Technology
Maulana Abul Kalam Azad University of Technology

Examination Roll No.: M4PRD22006
Registration No.: 154474

THESIS
SUBMITTED IN PARTIAL FULFILLMENT OF THE REQUIREMENT FOR THE AWARD OF THE DEGREE OF MASTER OF PRODUCTION ENGINEERING IN THE FACULTY OF ENGINEERING AND TECHNOLOGY,
JADAVPUR UNIVERSITY

DEPARTMENT OF PRODUCTION ENGINEERING
JADAVPUR UNIVERSITY
KOLKATA-700032

JADAVPUR UNIVERSITY
FACULTY OF ENGINEERING AND TECHNOLOGY

CERTIFICATE OF RECOMMEDATION

I HEREBY RECOMMEND THAT THE THESIS ENTITLED **"ONLINE PATH PLANNING OF A MOBILE ROBOT USING LASER RANGE SENSOR BASED ON AN IMPROVED E-BUG ALGORITHM"** CARRIED OUT UNDER MY/OUR GUIDANCE BY **MR. SOURAV ADHIKARY** MAY BE ACCEPTED IN THE PARTIAL FULFILLMENT OF THE REQUIREMENTS FOR THE DEGREE OF "MASTER OF PRODUCTION ENGINEERING".

_____

(**Dr. AJOY KUMAR DUTTA**)
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-700032

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 pre-requisite 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 conclusion drawn therein but thesis only for the purpose for which it has been submitted.

_____

COMMITTEE ON
FINAL EXAMINATION
FOR EVALUATION OF
THE THESIS

(External Examiner)

_____

(Internal Examiner)

ACKNOWLEDGEMENT

While bringing out this thesis to its final form, I came across a number of people whose contributions in various ways helped my field of project work and they deserve special thanks. It is a delight to convey my gratefulness to all of them. First and foremost, I would like to express my deep sense of gratitude and indebtedness to my supervisor Prof. Ajoy Kumar Dutta, Production Engineering Departmentat, Jadavpur University for his priceless and meticulous supervision at each and every phase of my work inspired me in innumerable ways. Special thanks to Mr. Subir Kumar Debnath, Associate 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.

_____

SOURAV ADHIKARY
Class Roll No: 002011702002
Examination Roll No.: M4PRD22006

TABLE OF CONTENTS

CHAPTER 1

1.0. INTRODUCTION

1.1 INTRODUCTION TO MOBILE ROBOTS

A mobile robot can be defined as an automatic machine that is capable of locomotion. Mobile robotics is usually considered to be a subfield of robotics and information engineering.

Mobile robots have the capability to move around in their environment and are not fixed to one physical location. Mobile robots can be "autonomous" (AMR - autonomous mobile robot) which means they are capable of navigating an uncontrolled environment without the need for physical or electro-mechanical guidance devices. Alternatively, mobile robots can rely on guidance devices that allow them to travel a pre-defined navigation route in relatively controlled space. By contrast, industrial robots are usually more-or-less stationary, consisting of a jointed arm (multi-linked manipulator) and gripper assembly (or end effector), attached to a fixed surface.

Mobile robots have become more commonplace in commercial and industrial settings. Hospitals have been using autonomous mobile robots to move materials for many years. Warehouses have installed mobile robotic systems to efficiently move materials from stocking shelves to order fulfillment zones. Mobile robots are also a major focus of current research and almost every major university has one or more labs that focus on mobile robot research. Mobile robots are also found in industrial, military and security settings.

The components of a mobile robot are a controller, sensors, actuators and power system. The controller is generally a microprocessor, embedded microcontroller or a personal computer (PC). The sensors used are dependent upon the requirements of the robot. The requirements could be dead reckoning, tactile and proximity sensing, triangulation ranging, collision avoidance, position location and other specific applications. Actuators usually refer to the motors that move the robot can be wheeled or legged. To power a mobile robot usually we use DC power supply (which is battery) instead of AC. A robot can be defined as ‘a mechanical device which performs automated tasks, either according to direct human supervision, a pre-defined program or, a set of general guidelines, using artificial intelligence techniques’. The first commercial robot was developed in1961 and used in the automotive industry by Ford. The robots were principally intended to replace humans in monotonous, heavy and hazardous processes. Nowadays, simulated by economic reasons, industrial robots are intensively used in a very wide variety of applications. Most of the industrial robots are stationary. They operate from a fixed position and have limited operation range. The surrounding area of the robot is usually designed in function of the task of the robot and then secured from external influences.

These robots efficiently complete tasks such as welding, drilling, assembling, painting and packaging.

Mobile robots have different characteristics like some are operated from fixed based and some aren't. Most stationary robots like industrial robots fixed at a stationary position and have a bounded operating range. They are programmable, automated but capable of movement three or more axes. They are used in painting, pick and place operation, assembly of automobiles, packaging and labeling, product inspection and testing all are done with high endurance, precision and speed. But instead of performing on a fixed location there are need to develop some awareness like environment interactions through sensors, on board intelligence for determination the best action to take and development of intelligent navigation system which ensures collision free movement.

Generally mobile robots are such types of robots which can autonomously move around environment. In mobile robots, mobility gives a much greater flexibility for performing versatile application field. Robots with mobility can complete a task easily even if it is not designed for environments. In mobile robots, there are locomotive mechanisms that help to move throughout its environment. There are large varies of approach to move and so that the robot's approach to locomotion is a very important aspect of mobile robot. In this approach a mobile robot can walk, jump, skate, swim, fly etc.

The mobile robots operate their operations using a combination of artificial intelligence (AI) and physical elements. Tracks, legs and wheels are the physical elements of the mobile robots. The usages of mobile robot are versatile in business sectors. The mobile robots are even accomplished such works which are dangerous or impossible for human workers.

1.2 OVERVIEW OF PATH PLANNING OF MOBILE ROBOTS

Path planning is one of the most crucial research problems in robotics from the perspective of the control engineer. Many problems in various fields are solved by proposing path planning. It has been applied in guiding the robot to reach a particular objective from very simple trajectory planning to the selection of a suitable sequence of action. Path planning cannot always be designed in advance as the global environment information is not always available a priori. By proposing a proper algorithm, path planning can be widely applied in partially and unknown structured environments.

An appropriate trajectory is generated as a sequence of actions to maintain the robot movement from the start state to the target point through several intermediate states. Every decision in path planning algorithms is selected according to the available information in the current state and used criteria such as the shortest distance measures to the target point using Euclidean distance computation. There may be more than one path from the start state to the target point. However, in several situations, there is no possible path to reach the goal states. In terms of optimization, the ideal path must be the shortest distance and far from obstacles/collision-free, and spend the shortest time to reach the goal state. Also, the selected trajectory must be smooth without extreme turns as a robot may have several motion constraints, such as the no holonomic condition in under actuated systems.

The nature of an obstacle is described via its configuration which may be convex shaped or concave shaped or both. The status of an obstacle may be static (when its position and orientation relative to a known fixed coordinate frame is invariant in time), or dynamic or orientation or both change relative to the fixed coordinate frame change.

Mobile robot path planning is an important research field of robotics. It refers to that, the mobile robot in a work environment with obstacles, based on one or some optimization criterion, search for a motion path from the initial state to target, state and the path is the optimal, near optimal, safe, obstacle avoidance. Robot planning is concerned with the general problem of figuring out how to move to get from one place to another and how to perform a desired task. As a whole it is a wide research in itself. Actually the term path planning means different things to different scientific communities. Planning represents a class of problem solving which is an interdisciplinary area of system theory and artificial intelligence. The objective of path planning for mobile robots to find a collision free path from a starting point

to target point and to optimize it with respect to certain criteria. Thus determining an optimal path is more difficult within the environments containing a number of obstacles.

Various methods of path planning will be discussed in details in chapter 2.

1.3 LITERATURE SURVEY

Many research work have been carried out in the field of mobile robot path planning. Some of those researches have been reported below:

Sariff et al. [1] presented an overview of autonomous mobile robot path planning focusing on algorithms that produce an optimal path for a robot to navigate in an environment. To complete the navigation task, the algorithms will read the map of the environment or workspace and subsequently attempts to create free paths for the robot to traverse in the workspace without colliding with objects and obstacles. Appropriate or correct and suitable algorithms will fulfill its function fast enough, that is, to find an optimal path for the robot to traverse in, even if there are a large number of obstacles cluttered in a complex environment. To achieve this, various approaches in the design of algorithms used to develop an ideal path planning system for autonomous mobile robots have been proposed by many researchers. In this paper they give a overview and discusses the strength and weakness of path planning algorithms developed and used by previous and current researchers.

Alajlan et al. [2] concentrated the matter on the collaboration of efficient multi-sensor systems for developing the optimal motion planning for mobile robots. The algorithm introduced produces the shortest and most energy efficient path from a reference point to its goal point. The distance travelled at the time taken in addition to the consumed energy have the asymptotic complexity of O (n log n) when is defined as the number of obstacles. Realtime experiments are conducted for the testing of its efficiency and accuracy.

Ng et al. [3] concluded that the bug algorithm family are well known for robotic navigation algorithms with proven termination condition for unknown environments. Eleven such variation has been implemented and compared on the Eyesim simulation platform. Bug algorithm performance depends greatly based on a given environment. Tangent bug produces the shortest trajectory with spaces that allows it to utilize its range sensors (environment a, c and d). Tangent bug was able to drive directly towards a vertex whereas other programs relied on parameters such as wall. The second shortest environment A path was achieved by Distbug as it is utilized the range sensors for immediate detection of the visible target. In environment B, Rev 2 produced the shortest path because the wall following strategy minimized the wall following paths, in environment C Distbug what's the shortest path user because its range sensor allows the robot to abandon an obstacle earlier. In environment D Leave bug and Bug

1 ranked equivalently for the second shortest path. As for implementation complexity subjective enrankment of bug algorithms from simple to complex as: Class 1, Bug2, Bug1, One bug, Alg2, Rev2, Alg1, Rev1 and ultimately tangent bug. Besides, simulations were conducted both in a perfect noise free environment as well as noisy spot aclcompanied by small errors in sensors.

Belkhous et al. [4] proposed a new phenomenon for trajectory optimisation of a mobile robot in a General dynamic environment. The new technique is the integration of the static and dynamic modes of trajectory planning to provide an algorithm that yields fast an optimal solution for static environment and produces a new path when an undesirable situation comes in. The particularity of this new technique is a representation of the static environment in a judicious manner facilitating the path planning and the reduction of processing period. Moreover, when an Unexpected obstacle blocks the robot trajectory the method uses the robot sensors to detect the presence of the obstacle and finds a best way to tackle it up and circumvent it and then proceeds towards final point. Experimental results are a source of evidence of its effectivity.

Zi-Xing et al. [5] involves the delineation of an inverse D* algorithm for the path planning in unknown circumstances. In this inverse D* algorithm the local potential energy around the current position is created for sleep by defining the robotic distance, then the leave point is searched to be regarded as the local goal position satisfies the requirement of the Rolling Optimization. The leave Point is searched locally and iteratively until the robot gets to its destination. Experimental evidences support the validity of the algorithm.

Langer et al. [6] improved the performance of the existing methods of path planning that utilise local information or as an entirely new method, if global information is available. It is also presented with a short comparison of methods found in Literature this giving a brief evidence of its efficiency, low computational cost and effective service even in Complex circumstances.

Roy et al. [7] presented 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 process through different image processing and machine learning techniques. Obstacles in the path of the robot is also tried to process 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.

Nguyen et al. [8] adopted and implemented a new path planning strategy for mobile robots On the other hand based on the shortest path from the initial point to the goal point This path planner can opt the best direction of the MR which helps to reach the destination as soon as possible. On the other hand, with an intelligent obstacle avoidance system this method could sort out the target point with the nearest and shortest path length while avoiding some infinite loop traps of various obstacle in undesirable circumstances. Diffusion of two approaches enables that an hour to reach the final point with a very reliable algorithm. Move by spontaneous improvements of the onboard sensor's information, this approach can generate the MR 's trajectory both in static and dynamic circumstances. A large number of simulation in some similar circumstances frankly illustrates the efficiency of the proposed path planning algorithm.

Ganeshmurthy et al. [9] proposed a heuristic based method to search the feasible initial path efficiently. The heuristic based method is then combined into the simulated annealing algorithm based approach for dynamic robot path planning. Therefore, the quality of the solution is characterized by the length of the planned path and it is improved with the combined heuristic method in the simulated annealing based approach for both runtime and offline path planning.

Alpaslan YU et al. [10] implemented on a Pioneer mobile robot on the simulation environment (Mobile sim). sensing elements involve the utilization of Sonar range sensors. This study demonstrates the mobile robots build a new motion planning using the bug algorithm only if their meeting is with an undesirable obstacle during in journey to the destination. Each of the bug algorithm is examined separately for a similar configuration space point. The study ends with the performance comparison of the burg algorithm.

Devi et al. [11] proposed that the principal objective of the robot is to search for collision Free environment in order to reach the specified point. The main challenge in the robotic navigation is the localization i.e the robot should know its current position. In this case the localization problem is sorted out by using graphical method. This algorithm could be included in the bug

algorithm family. Dynamic point bug algorithm has been implemented for robot navigation system in NI - sbRIO 9631. The proposed work offers solution to identify the current position of the robot while proceeding towards the target.

Al-Jarrah et al. [12] presented a noval detection algorithm for vision system which is the combination of fuzzy image processing and bacterial algorithm. Main target is to increase the detection efficiency and reduce the computational time. 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. 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.

Han et al. [13] generate a collision-free path from a starting position to a target position with respect to a certain fitness function, such as distance. They propose a new methodology to solve the path planning problem in two steps. First, the surrounding point set (SPS) is determined where the obstacles are circumscribed by these points. After the initial feasible path is generated based on the SPS, they apply a path improvement algorithm depending upon the former and latter points (PI FLP), in which each point in the path is repositioned according to two points on either side. Through the SPS, it is able to identify the necessary points for solving path planning problems. PI FLP can reduce the overall distance of the path, as well as achieve path smoothness. The SPS and PI FLP algorithms were tested on several maps with obstacles and then compared with other path planning methods As a result, collision-free paths were efficiently and consistently generated, even for maps with narrow geometry and high complexity.

Zohaib et al. [14] 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. Simulation results of robot navigation in an environment with obstacles demonstrate the performance of the improved algorithm.

Borenstein et al. [15] introduced and implemented a new real-time obstacle avoidance system for mobile robots. This system permits the detection of unknown obstacle simultaneously with the navigation of the mobile robot to avoid collision and proceed towards the target. The upgradation of the system which is entitled as virtual force field lies in the integration of two know concepts: the uncertainty grids for the obstacle representation and the potential field for the robotic navigation. The combination is especially suitable and proves to be utilitarian for the accommodation of inaccurate sensor data (such as produced by LASER sensors) as well as for sensor fusion and this enables continuity of the robot without letting it stop its motion in front of obstacles. This navigation also never fails to take under consideration the dynamic Behavior for fast mobile robot and solves a local minimal trap problem. Conclusion from experimental results from a mobile robot running at a maximum speed of 0.78 m/s clearly demonstrates the power of the proposed algorithm.

Harshini et al. [16] implemented a diverse approaches and techniques to solve a path planning problem by considering certain factors like obstacle shape, its orientation, type of environment etc. Based on the surrounding environment the robot navigates globally or locally. This paper focuses on the navigation of mobile robot operating in a static environment consisting of elliptical and polygonal obstacles. A mathematical formulation has been developed to obtain these paths and also to find the shortest path among them using Centre of Gravity Approach (CGA) and Coordinate Reference Frame (CRF) technique. The simulation results prove the proposed approach to be effective as the robot navigates to the defined target point without colliding with the obstacles in the environment.

Sankaranarayanan et al. [17] considered amidst unknown obstacles in a two dimensional plane. Using only the local information like MA's current position and whether it is in contact with an obstacle. The two algorithms which solve this problem are due to Lumelsky and Stepanov. They proposed a new path planning algorithm to solve this problem. The motivation for the new algorithm is to realize the smallest worst case path length possible in its category. The procedure for the new algorithm is presented with explanations. Its various characteristics

like, local cycle creation, worst case path length, target reach ability conditions, etc, are dealt with. Its performance is compared with those of the existing algorithms.

N Buniyamin et al [18] presented a path planning algorithm for an autonomous robot. They mainly focused on the bug algorithm family which is a local path planning algorithm. Bug algorithms use LASER sensors to detect the nearest obstacle as a mobile robot moves towards goal with limited information. The algorithm uses obstacle border as guidance toward the target as the robot circumnavigates the obstacle till it finds certain condition to fulfill the algorithm criteria to leave the obstacle toward target point. introduces an approach utilizing a new algorithm called PointBug. This algorithm 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 generate 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.

MandaI et al. [19] presented an algorithm for path planning to a target for mobile robot in unknown environment. The proposed algorithm allows a mobile robot to navigate through static obstacles and finding the path in order to reach the target without collision. This algorithm provides the robot the possibility to move from the initial position to the final position (target). The proposed path finding strategy is designed in a grid-map form of an unknown environment with static unknown obstacles. The robot moves within the unknown environment by sensing and avoiding the obstacles coming across its wcry towards the target. When the mission is executed, it is necessary to plan an optimal or feasible path for itself avoiding obstructions in its way and minimizing a cost such as time, energy and distance. The proposed path planning must make the robot able to achieve these tasks: to avoid obstacles and to make ones wcry toward its target. The algorithms are implemented in Matlab, afterwards tested with Matlab GUI; whereby the environment is studied in a two dimensional coordinate system.

Hachour et al. [20] presented an algorithm for path planning to targeted point for mobile robot in unknown environment. The proposed algorithm enables robot to circumnavigate through static obstacles and finding the path in order to approach the destination avoiding collisions. This algorithm provides the robot the probability to move from the initial to the target. The introduced path finding strategy is developed in a grid map form of an unknown environment

with static unknown obstruction. The robot displaces itself within the unknown environment by sensing and avoiding the obstructions coming across its path towards the target. When the set is executed it is vital to plan and optimal of feasible path for itself avoiding obstacles in its way and minimising factors such as cost, time, energy as well as distance. The proposed path planning must ensure that the robot is able to achieve this tasks: to avoid obstacles, and to make its way toward its target. The algorithms are implemented in Borland C++, afterwards tested with Visual Basic and DELPHI programming language, whereby the environment is studiied in a 2D coordinate system. The simulation part is a very nice approach to the actual expected result. This part is done using C++ to recognise all objects within the environment and since it is suitable for graphical problems. Combining the segmented environment issued from C++ development, the algorithm allows the robot to get displaced to the desired location following an estimated trajectory using Visual Basic and DELPHI language.

Iijima et al. [21] introduced an efficient approach that beholds the geometric world map of the unknown 2-D environment where the mobile robot gets displaced by searching it using range sensors included in the mobile robot. In this searching method the robot observes and decides the observation points and acquires the information of its surrounding VIT the range sensors. According to this information the range sensors will specify the boundary between the floor and walls around it. Then the surrounding model could be generated by acquiring spontaneously the extracted boundary information at each observation spot meanwhile applying a suitable adjustment algorithm on them. In order to illustrate the process of the map construction using the proposed algorithm a simulator based on the actual mobile robot is presented. The simulation results clearly show the effectiveness of the proposed algorithm in the real circumstances considering the sensory and control errors.

Contreras-Cruz et al. [22] approached to sort out the mobile robot path planning problem is introduced.This section combines the artificial Bee Colony algorithm as a local search procedure and the evolutionary programming algorithm to find the feasible path found by a set of local procedures. The proposed process is compared to classical probabilistic road map method (PRM) with relative to their planning performances on a set of benchmark problems and it exhibits a better performance. Criteria that is implied to measure planning efficiency involves the path length, the smoothness of path, the computation period and the success rate of planning. Experimental results including statistical data illustrates the significance of the proposed model.

Kuric et al. [23] presents the methods of navigation and mapping of mobile robots in an indoor environment, for instance laboratories, buildings, corridors and so on. It delineates the proposed solution of global navigation in more detail through the implication of potential field method and its transformation into the topological Map. Two separate software tools were designed for simulation of wheeled mobile robot behaviour in the authors workplace. The first software involves the utilisation of the metric form of space representation and it can simulate tactic level of global navigation while, the second one deals with its transformation into the topological map and could be utilised for strategic level of global circumnavigation. This is how it is possible to get the so-called multilayer mat system suitable for various tasks of robot navigation and path planning.

Cai et al. [24] 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 branchand-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.

Jin et al. [25] proposed an algorithm for planning an optimal path to capture a moving object by a mobile robot in real-time. The direction and rotational angular velocity of the moving object are estimated using the Kalman filter, a state estimator. It is demonstrated that the moving object is tracked by using a 2-DOF active camera mounted on the mobile robot and then captured by a mobile manipulator. The optimal path to capture the moving object is dependent on the initial conditions of the mobile robot, and the real-time planning of the robot trajectory is definitely required for the successful capturing of the moving object. Therefore, the algorithm that determines the optimal path to capture a moving object depending on the initial conditions of the mobile robot and the conditions of a moving object is proposed in this paper. For real-time implementation, the optimal representative blocks have been utilized for the experiments to show the effectiveness of the proposed algorithm.

Pandey et al. [26] introduced a singleton type-1 fuzzy logic system (T1-SFLS) controller and Fuzzy-WDO hybrid for the autonomous mobile robot navigation and collision avoidance in an unknown static and dynamic environment. The WDO (Wind Driven Optimization) algorithm is used to optimize and tune the input/output membership function parameters of the fuzzy controller. The WDO algorithm is working based on the atmospheric motion of infinitesimal small air parcels navigates over an N-dimensional search domain. The performance of this proposed technique has compared through many computer simulations and real-time experiments by using Khepera-III mobile robot. As compared to the T1-SFLS controller the Fuzzy-WDO algorithm is found good agreement for mobile robot navigation

Leena et al. [27] Autonomous navigation of mobile robots is an area that has witnessed a lot of research activity in the recent years due to its increasing applications. Several approaches have been proposed for the navigation of mobile robots. This review paper describes the various developments and techniques that have been applied for navigation of robots in dynamic environments with special focus on the soft computing approaches.

Sahu et al. [28] intended to obtain safe near-optimal path for single and multiple mobile robots in static global environment. The computational mobile robot path planning is done by Genetic algorithm with enhanced fitness function. The fitness function considered safety along with collision free and shortest path. Results obtained for different safety parameter coefficients are compared. It is observed that increase in safety achieves in expense of extra path length. In any application safety is major and prime factor to be considered; in this paper safety parameter is used and shows the effect of safety parameter on optimality and path length.

Akka et al. [29] introduced 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.

Haj Darwish et al. [30] presented 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 subpaths. 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 ActivityBot 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.

Li et al. [31] proposed a self-adaptive learning particle swarm optimization (SLPSO) with different learning strategies. First the path planning problem has been transformed into a minimization 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). 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.

In the present work, online path planning of a mobile robot system has been considered in which a near optimal path from a start position to a goal (target) position has been obtained while avoiding any collision between the mobile robot and obstacles in its path which are detected by a LASER range sensor using modified E-Bug algorithm, where the actual dimension of the mobile robot has been considered instead of considering the robot to be a point.

1.4 OBJECTIVES AND SCOPE OF PRESENT RESEARCH WORK

A Parallax mobile robot (ActivityBot), installed in the Robotics Laboratory of Production Engineering Department of Jadavpur University, has been used for the present project. Main aim of the present work is the development of necessary algorithm and software to navigate a mobile robot in presence of obstacles from start point to destination point using a LASER range sensor, mounted on the Activity Bot, to avoid any obstacle in its path. The ActivityBot robot will move towards the destination (goal) from start position until it meets an obstacle which is detected by the LASER range sensor. For this present work, a local path planning has been used based on modified form of E-Bug algorithm.

For moving the robot forward along the path of target point, next point from any current point is determined by E-Bug algorithm by the output of range sensor which gives the distance of nearest obstacle from the sensor and also detects the sudden points (sudden change in distance from the sensor to nearest obstacle). The reading of distance either increasing or decreasing by a considerable amount according to the sudden change of range sensor output is considered for detecting the sudden points. The details of the algorithm and developed program will be discussed in chapter 2 and chapter 4 respectively.

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

a. To set up an arrangement for the workspace consisting of ActivityBot robot fitted with a LASER range sensor and obstacles on a suitable worktable having a marked boundary.

b. To mount the LASER, range sensor system on the ActivityBot mobile robot system, and make necessary hardware connections to connect it to the propeller microcontroller through its input-output port (pins). To make necessary arrangements to be made to rotate the LASER sensor through 180° by a servo motor fitted to the ActivityBot mobile robot system.

c. To make necessary arrangement to orient the LASER sensor so that the axes of its emitter and receiver are in the same vertical plane for getting reflected waves symmetrically from both sides of any object.

d. To develop a simplified algorithm based on E-Bug algorithm with necessary modification for considering the actual dimension of the ActivityBot mobile robot for moving the mobile robot from a start point to a target point by detecting and avoiding obstacles, if any, along its path towards goal.

e. To develop a program in Propeller C language using Simple IDE software for the ActivityBot mobile robot for producing necessary movements of the robot in presence of static obstacles for moving from a starting point to a target point using modified E-Bug algorithm, as mentioned in (d).

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

Scope

The analyses, techniques and algorithms described in the thesis are related to any mobile robot, and the program development and experiments described in the thesis are related to a particular Parallax ActivityBot mobile robot fitted with a LASER range sensor and run by Simple IDE software existing in the Robotics Laboratory of Production Engineering Department of Jadavpur University.

CHAPTER 2

2.0. PATH PLANNING METHODS FOR MOBILE ROBOTS

2.1 TYPES OF PATH PLANNING METHODS

The basic requirement for mobile robots in most applications is the ability to navigate form a start point to a given goal point along a collision free path that must be generated from the start point to the given goal point. It has been applied in guiding the robot to reach a particular objective from very simple trajectory planning to the selection of a suitable sequence of action. Path planning cannot always be designed in advance as the global environment information is not always available a priori. By proposing a proper algorithm, path planning can be widely applied in partially and unknown structured environments.

Accurate path planning enables autonomous mobile robots to follow or track an optimal collision free path from start position to the goal position without colliding with obstacles in the workspace area. An ideal path planner must be able to handle uncertainties in the sensed world model, to minimize the impact of objects to the robot and to find the optimum path in minimum time especially if the path is to be negotiated regularly. Many path planning algorithms have been developed over the years for the navigation of mobile robots. These algorithms may be generally classified into different categories that are now described briefly.

2.1.1 Path Planning in Static and Dynamic Environment

In general dynamic environment there is a new optimization technique of mobile robot which combines static and dynamic mode of trajectory planning to provide an algorithm that gives fast and optimal solutions for static environments, and generates a new path when an unexpected situation occurs. The most conducted works in the trajectory optimization domain deal with either static known environments or dynamic environments where the mobile robot is confronted with unknown obstacles.

In **static environment** the techniques are able to find the shortest path between two points but are ineffective when an unexpected obstacle comes to block the robot's trajectory [12].

In **dynamic environments** (an unforeseen obstacle blocking the robot's trajectory) the robot is unable to pursue its path conveniently. The optimization of a traveling mobile robot's path, in dynamic environments, has been the purpose of many investigations. The dynamic algorithms allow reaching the desired destination in an unknown environment but are very slow and are not conceived to find the optimal path. A natural extension to the basic path

planning problem is planning in dynamic environments Dynamic environment contains moving obstacles as well as stationary obstacles.

2.1.2 Local and Global Path Planning

Local path planning is path planning that requires robot to move in unknown environment or dynamic environment where the algorithm is used for the path planning will response to the obstacle and the change of environment. Local path planning also can be defined as real time obstacle avoidance by using sensory based information regarding contingency measures that affect the save navigation of the robot [32]. In local path planning, normally, a robot is guided with one straight line from starting point to the target point which is the shortest path and robot follows the line till it senses obstacle. Then the robot performs obstacle avoidance by deviating from the line and in the same time update some important information such as new distance from current position to the target point, obstacle leaving point and etc. In this type of path planning, the robot must always know the position of target point from its current position to ensure that robot can reach the destination accurately. Potential field method [33] is the one of the well-known local path planning technique.

Global path planning is a path planning that requires robot to move with priori information of environment. The information about the environment first loaded into the robot path planning program before determining the path to take from starting point to a target point. In this approach the algorithm generates a complete path from the start point to the destination point before the robot starts its motion [34]. Global path planning is the process of deliberatively deciding the best way to move the robot from a start location to a goal location. Thus for global path planning, the decision of moving robot from a starting point to a goal is already made and then robot is released into the specified environment.

2.2 OVERVIEW OF DIFFERENT PATH PLANNING METHODS

The motion planning approaches can be divided into conventional and biologically inspired methods. Some methods for robot motion planning are as follows:

2.2.1 Graph Searching Technique

The purpose of the graph search is to efficiently produce a set of paths that explore all regions in the configuration space and that may contain the optimal path. Using the graph search algorithm, the optimal path is the one that can be traversed at the minimum time between given end points. One approach to generating a large set of paths, using a graph search, is to use the *k*-best search by Dreyfus [35] to produce a set of shortest paths. It is similar to a shortest path search except that it effectively excludes the $k - 1$ best paths from the searched space while searching for the next k^{th} best path.

2.2.2 Artificial Potential Fields

The artificial potential field (APF) based path planning methods have a local minimum problem, which can trap mobile robots before reaching its goal. The artificial potential field approaches are much convenient than Artificial Intelligence methods for their high efficiency in path planning provided that the working environment is known. Artificial Potential Field (EAPF) is proposed for real-time robot path planning. The artificial potential field method is combined with genetic algorithms, to derive optimal potential field functions. The proposed Evolutionary Artificial Potential Field approach is capable of navigating robot(s) situated among moving obstacles.

2.2.3 Soft Computing Technique

Soft computing techniques are a new class of techniques which is a combination of advanced theories and technologies such as Neural Network, Fuzzy Logic, Genetic Algorithm and Probabilistic reasoning. Soft computing technique is made based on human brain ability which is able to do difficult and complicated works. Soft computing includes not only the previously mentioned approaches but also useful combinations of its components. Soft computing includes Neuro Fuzzy Systems, Fuzzy Neural Systems, and usage of Genetic Algorithms in Neural Networks and Fuzzy Systems and many other hybrid methodologies.

The main characteristic of soft computing is its "adaptive behaviour" so that systems adapt to users' perceptions [3]. Some soft computing techniques are given below:

- Neural Network
- Fuzzy Logic
- Genetic Algorithms
- Ant Colony Optimization
- Hybrid Methods

2.3 BUG GROUP OF ALGORITHMS

The Bug algorithms are simple algorithms which are well known mobile robot navigation methods for local path planning with minimum sensor without generating a full map of the environment. In these algorithms the robot takes an action on the basis of current percepts of sensors without taking into account the previous path and actions. The aim of the Bug algorithms is to guide a robot starting from a source (S) to a target (T) given that the robot has no knowledge of the complete environment. The robot should achieve this goal with as little global information as possible. When they face an unknown obstacle, they are able to easily produce their own path contouring the object in the 2D surface if a path to the goal exists. The purpose is to generate a collision-free path by using the boundary-following and the motion-to-goal behaviors. In addition, the Bug's family has three assumptions about the mobile robot: i) the robot is a point, ii) it has a perfect localization, and iii) its sensors are precise [19].

The following algorithms from the Bug family have been implemented and evaluated: Bug1, Bug2, Alg1, Alg2, Dist Bug, Class1, Rev1, Rev2, One Bug, Leave Bug and Tangent Bug.
This is not a complete list of Bug algorithms in existence. There are more examples VisBug21, VisBug2, HD1, Ave, Rover Bug, Wedge Bug, Cautious Bug etc. Those were not included for brevity and because they are quite similar to some of the included algorithms. For instance, Rover Bug, Wedge Bug and Cautious Bug are similar to Tangent Bug [11]. The variations of bug algorithms showed the effort toward shorter path planning, shorter timing, simpler algorithm and better performance. Langer, Coelho and Oliveira note that there is an increasing need for path planning algorithms in unknown environments for manufacturing, transport, goods storage, medicine (remote controlled surgery), military applications, computer games and spatial exploration. They simulated K-Bug in an office like environment. Bug algorithms were the first non-heuristic algorithms for motion planning in an unknown environment which guaranteed termination. Further, the robot does not need to build a map of the environment, it only needs to store one point for termination to be guaranteed. This makes the Bug algorithms highly suitable for real-time implementation. Lumelsky and Stepanov [36]. later extended this work to include range sensors. Some bug algorithms are described given below.

2.3.1 Bug1 Algorithm

The Bug1 algorithm was the first algorithm in the bug family created by Lumelsky and Stepanov[36]. Its principle is to advance toward the target along straight line until encountering an obstacle or reaching the target. When an obstacle is encountered the robot follows its boundary until encountering the hit point again, it means after its one complete circle, it restarts its motion around obstacles until reaches to leaving point. Then, the robot goes to the nearest boundary's point to the target and continues its path towards the goal.

In the Bug1 Algorithm the robot follows the following steps:

- The robot heads towards the destination.
- If the robot encounters an obstacle, it circumnavigates it and stores/remembers at each point how close it was to the goal.
- After circumnavigating the obstacle, the robot returns to the point that is closest to the obstacle by wall-following and then continues on the path towards the destination.

  The Bug1 algorithm is shown in Fig: 2.1

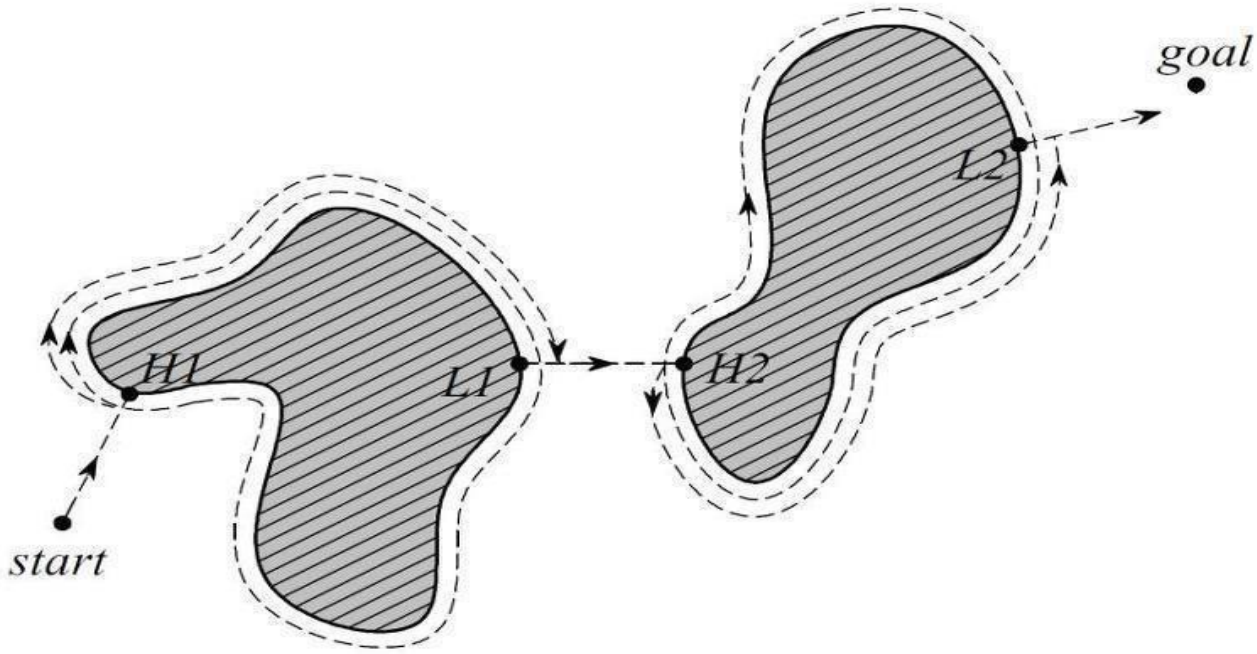


Fig: 2.1 Path generated by Bug1 Algorithm

2.3.2 Bug2 Algorithm

Bug2 algorithm is almost similar to Bug1 algorithm. But in this algorithm the mobile robot doesn't follow the whole boundary of obstacle. It follows the obstacle edge and calculating a new slope from every new position until the new slop matches the original slope [37]. When this slope becomes equal to slope of initial path, the behavior of the robot is changed to move to goal. Therefore, the robot follows single non-repeated path throughout its trajectory. Bug2 algorithm is more efficient than Bug1 algorithm as it allows the robot to reach the destination in less time following a short trajectory. The Bug2 algorithm is shown in Fig: 2.2

In the Bug2 algorithm the robot follows the following steps [38]:

- The robot heads towards the destination on the m-line which is defined as the line joining the start location and the destination.
- If an obstacle is encountered, the robot follows it until the time it again encounters the mline again closer to the goal.
- The robot then leaves the obstacle and again continues toward the goal on the m-line

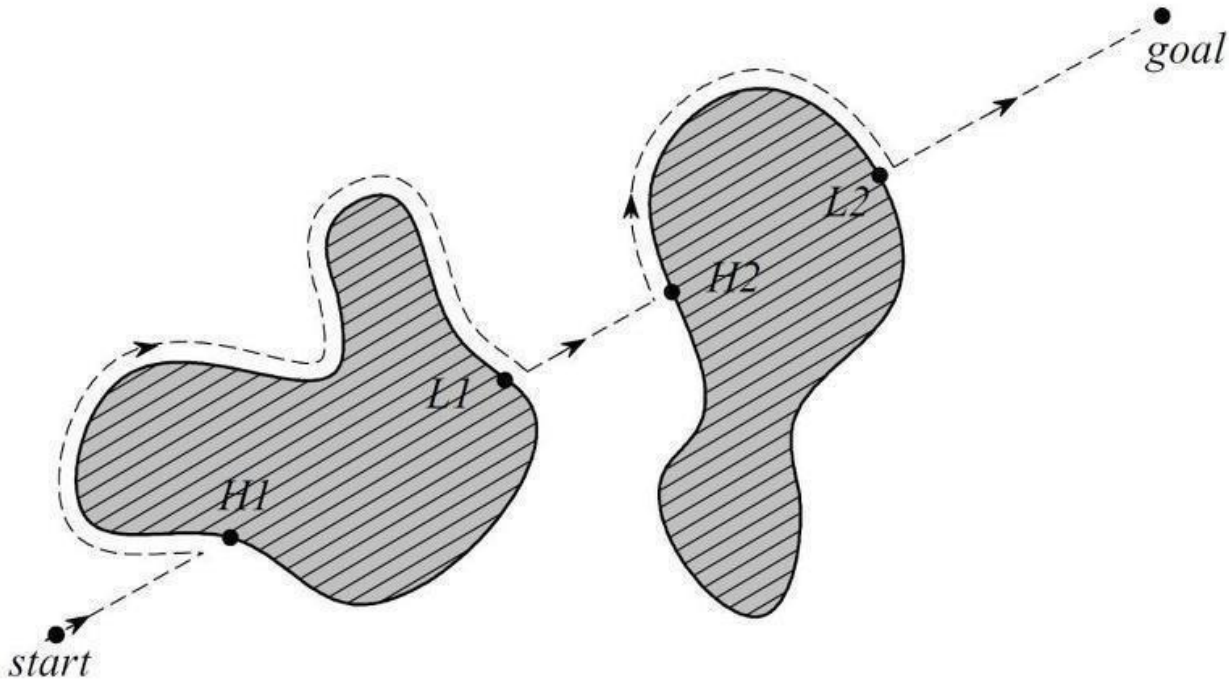


Fig: 2.2 Path generated by Bug2 Algorithm

2.3.3 Tangent Bug Algorithm

Kamon, Rivilin and Rimon introduce the Tangent Bug algorithm which is a range sensor based navigation algorithm for autonomous robots with two degrees of freedom. This algorithm uses the range sensor data to compute a locally shortest path based on the local tangent graph. Tangent Bug is one of the most frequently used algorithm for obstacle avoidance for sensor based mobile robots. In presence of static obstacles, the Tangent Bug algorithm can be used to plan a path from the start to the goal [11].

Tangent Bug algorithms principle is to advance towards target in straight line, and moving around encountered obstacles. The robot stops moving around obstacle and continues its path towards target once it founds a point on local tangent graph (LTG) closer to target then boundary's one. One advantage of tangent bug algorithm is that there is no need for wall following since the robot only turns on the spot and travels in straight lines. The generated path of Tangent bug algorithm as shown in Fig: 2.3.

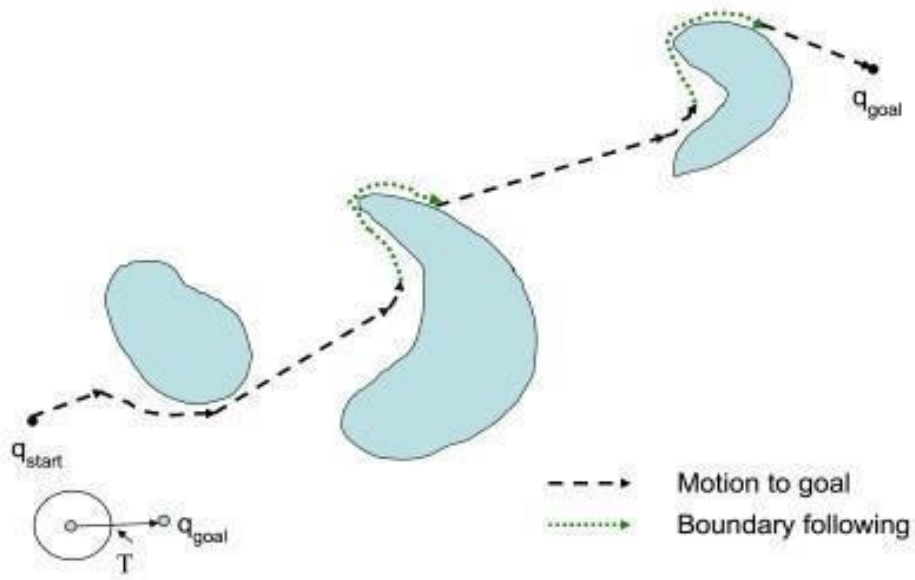


Fig: 2.3 Path generated by Tangent Bug Algorithm

2.3.4 Point Bug Algorithm

Point Bug algorithm has been implemented for robot navigation system [19]. This algorithm provides solution to identify the present location of the robot while moving towards target based on coordinates estimation. Point Bug algorithm is included in the Bug algorithm family. The point bug algorithm allows the robot to navigate in the given environment and it will avoid obstacle while it travels towards the target. The robot is equipped with an ultrasonic range sensor which helps in detecting an obstacle. The next point to move towards target is determined by the output of the range sensor which detects the sudden change in distance from sensor to the nearest obstacle.

It introduces the new concept of 'sudden point' which is a point where a sudden change (either increasing or decreasing) in distance of sensor's range is detected. The robot will move towards a sudden point according to the straight line (L) joining the current position and the target position. In the beginning it will move towards the target. When an obstacle is detected, the robot will avoid the obstacle and a new next position will be obtained. The flow chart of Point Bug algorithm is shown in Fig: 2.5

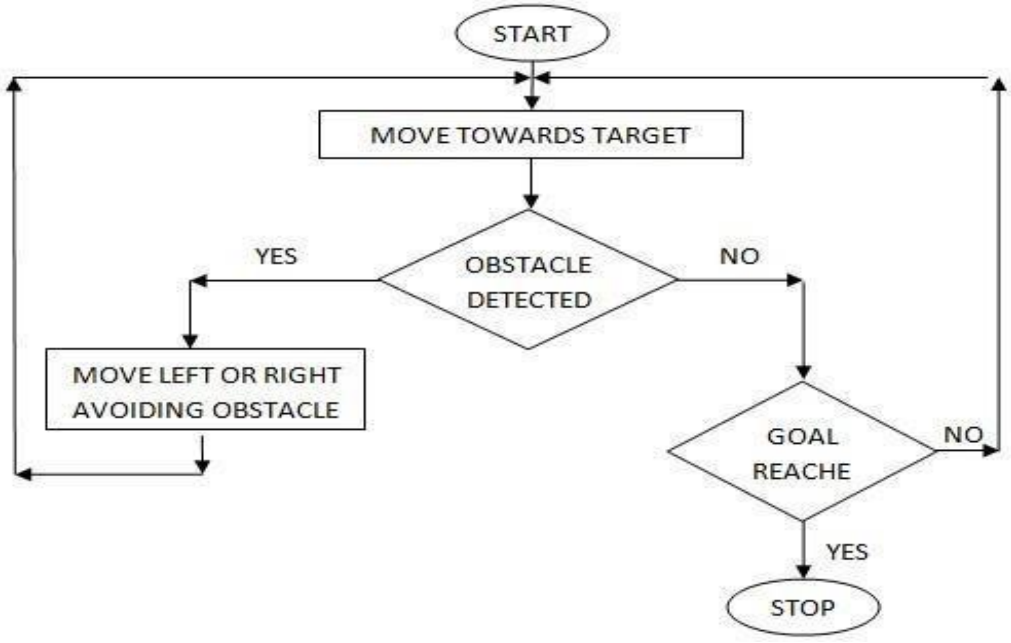


Fig: 2.5 Flow Chart of Point Bug Algorithm

Point Bug Algorithm Limitations

There are few limitations in Point Bug algorithm, some of which are given below [37]

- There are no tests in Point Bug algorithm if a sudden point has been already treated, since it does not record visited sudden points. This may result in producing an infinite loop.
- The choice of the point which minimizes the angular deviation relative to target direction does not produce the optimal path, as illustrated in Fig: 2.6. The deviation angle formed by point A is greater than that formed by point B, through the path length from S to D via point A is less than the path via B.

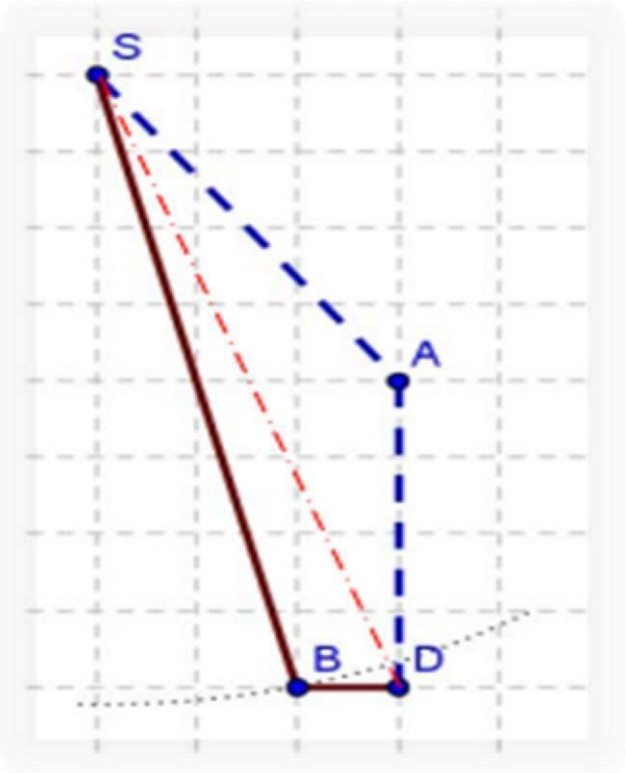


Fig: 2.6 Minimum angular Deviations does not guarantee optimal path

2.3.5 E-Bug Algorithm

Euclidian Bug is known as E-Bug Algorithm. E-Bug is a novel sensor-based path planning since it uses the Euclidian distance to choose the shortest path. E-Bug Algorithm is modification form of Point Bug Algorithm. E-Bug Algorithm's principle as Point Bug Algorithm [37]. The objective is to minimize the sum of sub-paths leading to the target. The main objective of E-Bug algorithm is to minimize the sum of existing sub-paths to find the near optimal path in an unknown environment. As the robot has very limited knowledge about the environment, it only knows the local sub-paths. So only the local sub-paths can be minimized for the successful implementation of the algorithm. Arriving at any sudden point, the robot chooses the shortest sub-path in the visible part of the environment, and assumes there doesn't exist an obstacle in the invisible part of the environment, until reaching this part. The proposed algorithm guarantees that the current sub path is optimal. In spite of limited sensor range and partial knowledge of the environment, EBug provides a good results comparing with other algorithms.

Description of E-Bug Algorithm

The optimal path never contains a cycle. A point can't exist twice in the optimal path. Three sets of points *(S*), (*C*) and (*R*) are considered as

- (**S**) is the set of all existing points (can perceived by the robot).
- (**C**) is the set of points formulate the optimal path (which considered as optimal).
- (**R**) is the set of rejected points.

Each time when the robot reaches a new sudden point the algorithm constructs a list of next sudden points. Then the robot chooses the point that generates the minimal path length. The sudden point that provides the minimal sum of distances will be elected. If a point is attained for a second time, all points encountered between these two detections will be moved to reject set (R). If the robot can't advance or detect any new sudden points from a given point, this point will be moved to the rejected set (R) and the robot returns to the previous sudden point.

E-Bug Algorithm Steps

Begin

1. From the current position enumerate all sudden point as detected by the range sensor.

If it reaches the goal end with success.

If no new sudden point is detected (C=ϕ) or all detected points are rejected (S=R). If the current position is the starting point.

Stop (the goal is not reachable).

Else go to the previous sudden point and move the current point to the rejected set (R)
Else for each detected sudden point expect the rejected ones, calculate the sum of

the two distances from the current position to this point and from this point to the

target. Go to point that provided minimal distance. End if

Go to 1

End

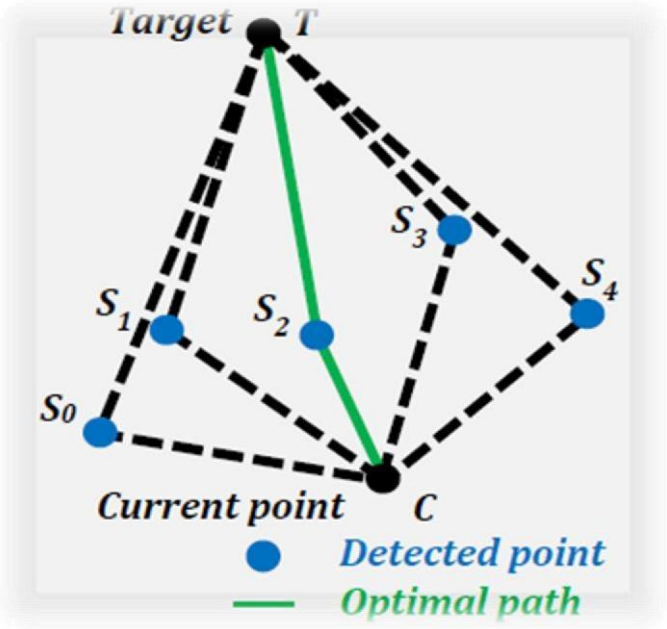


Fig: 2.8 The strategy of E-Bug Algorithm

The Fig:2.8 shows the strategy of E-bug algorithm. The algorithm calculates the length of all the sub-paths [C, S0] and [S0, T], [C, S1] and [S1, T], [C, S2] and [S2, T], [C, S3] and [S3, T], [C, S4] and [S4, T]. The robot will choose point S2 since the path length is minimal. E-Bug algorithm will test all accessible sudden points and finally choose the point which provides

shortest path to the target after the robot reaches at any sudden point which will be treated as current shortest point (C). The S2 point will be added to the current shortest point (C). When S2 is already selected, it will be added to the rejected set (R) and ignored afterwards. If no sudden point is detected further, the robot turns back and continues with the next one after adding current point to the rejected set (R). Thus the algorithm provides guarantee of testing of all accessible sudden point and termination of the algorithm is assured.
The robot chooses the shortest sub-paths in the visible (known) part of the environment after reaching any sudden point and thereafter assumes that no obstacle is present in the invisible (unknown) part of the environment that exist between the last sudden point and the target, until the goal is being reached. Therefore, the E-bug algorithm guarantees that the current sub-path is optimal.

In the present work, the basic concept of considering the sub-path with minimal length as used in E-Bug algorithm has been used with the following modifications:

- Instead of considering the mobile robot as a 'point', the actual dimension of the mobile robot has been considered.
- Instead of considering a sudden change (increase or decrease) in distance of sensor's range as 'sudden point' 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 dimension has been considered as a 'sudden point' for finding the sub-paths.

Several changes in the algorithm (and also in the program) have been made for incorporating these modifications and will be described in Chapter 4.

CHAPTER 3

3.0. SYSTEM HARDWARE AND SOFTWARE USED IN THE PROJECT

3.1 SYSTEM COMPONENTS USED IN THE PROJECT

The main components used in the present project can be divided into two parts hardware and software. The required components are mentioned below [42]:

- Parallax Activity Bot mobile robot kit, installed in the Robotics laboratory of Production Engineering Department of Jadavpur University.
- A LASER range sensor (LIDAR Lite V3) for providing the distance of the closest object from the sensor.
- Simple IDE software (Version 1-1-0) and Propeller C language to run the mobile robot.

Both the wheels of the mobile robot are equipped with incremental type optical encoders for providing the amount of rotation of the wheels. A number of obstacles of different shapes have been used for experimentation.

3.2 ACTIVITYBOT HARDWARE

The ActivityBot mobile robot is a compact, zippy robot which consists of a multi-core Propeller microcontroller along with great hardware: Versatile Propeller Activity Board perched atop classic, sturdy aluminum chassis, Optical encoders and wheels with secure Oring tires ensure straight straight-aways and consistent maneuvers, SD card for data logging and file storage, Electronic components for building navigation systems using touch, visible light, infrared light, and ultrasonic sensors. The Activity Bot is an ideal option for beginner robot builders and also a good introduction to the technology for use in schools and colleges. It allows users to learn engineering skills which have useful applications in the real world. It is easy to program using a Windows, Linux or Mac system. The complete robot kit includes a Propeller control board that makes it simple to integrate sensors, motors and more to make the various projects. It also has high-speed servo motors with optical encoders to provide fast, consistent and controllable movement. A sample view of the ActivityBot (fitted with LASER sensor) is shown in Fig 3.1.

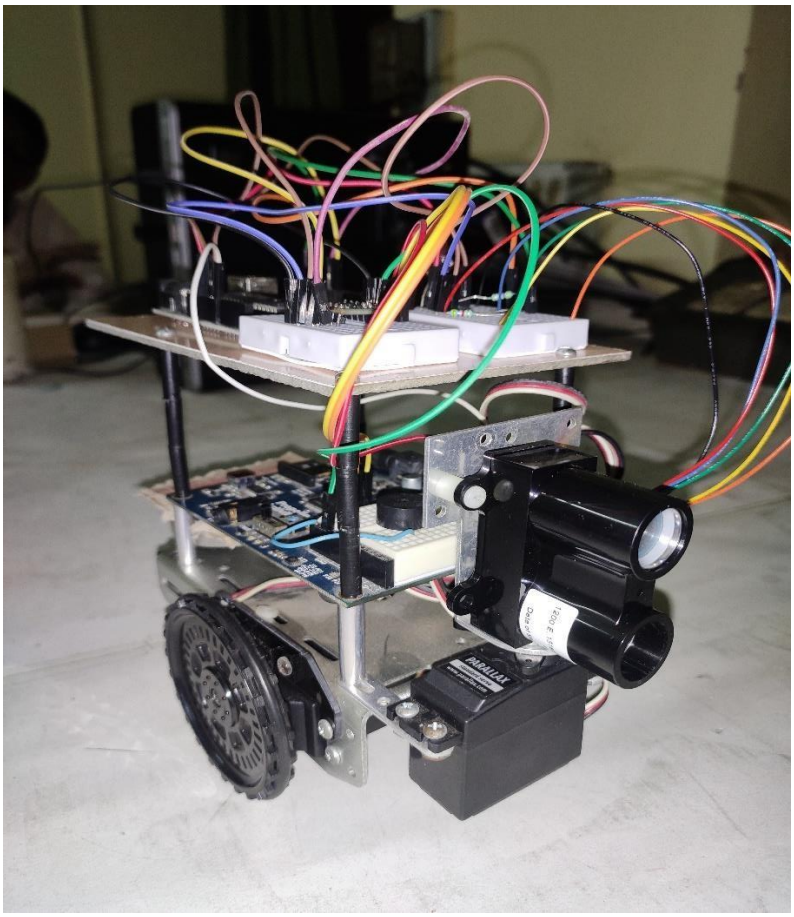


Fig: 3.1 Parallax Activity Bot fitted with LASER sensor

3.2.1 Propeller Activity Board

Propeller Activity Board features the 8-core Propeller programmable microcontroller prewired to a host of popular peripherals for fast processing of the applications. Programming is done through an easy to learn C language using software named Simple IDE.

Main Features of the Propeller Activity Board

- Built-in 8-core Propeller P8X32A microcontroller, 64 KB EEPROM, and 5 MHz crystal oscillator
- 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 overcurrent protection
- For external power supplies 6–15 V center-positive 2.1 mm barrel jack
- Reset button and 3-position power switch
- Onboard mini stereo-audio jack with microphone/video pass-through
- Built-in microSD card slot for data logging or storing WAV files
- XBee wireless module socket simplifies advanced applications
- 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.
- USB communication activity 3.3 V and 5 V switching voltage regulators with independent 1.8 amp outputs.

Key Specifications of the Board

- 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).

The details of Propeller Activity Board are shown in Fig 3.2.

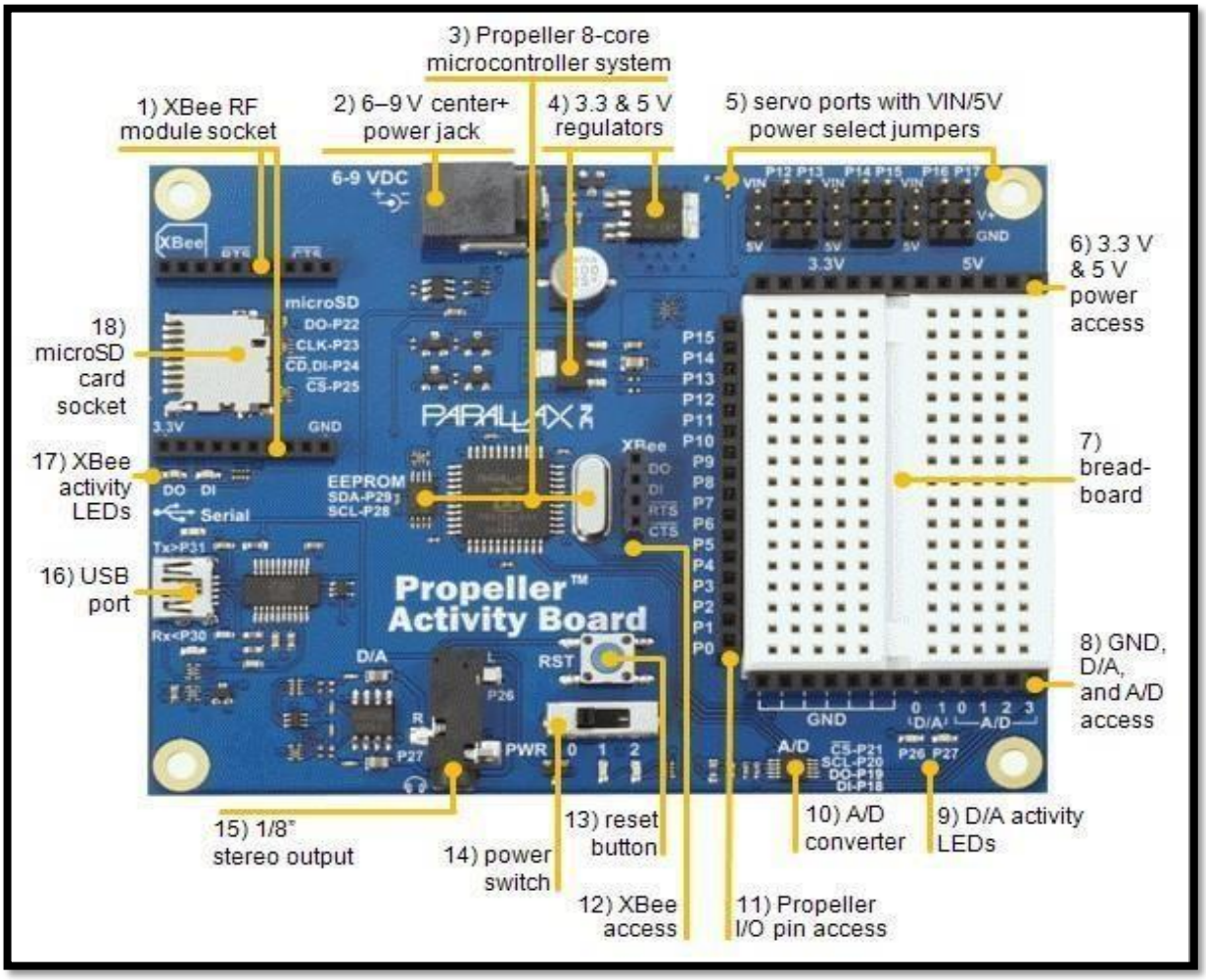


Fig: 3.2 Details of Propeller Activity Board

Briefly Description of Different Parts of Activity Board

In parallax Activity Board there are 18 different parts which are briefly described below –

1. XBee RF Module Socket

This socket fits most XBee wireless modules and is useful when the Propeller Activity Board needs to be part of a wireless network. Use it for robot control and robot team sports, remote data logging, and wireless message exchange with computer connected to another XBee module.

2. Power Jack

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. The 2.1 mm center positive power jack is one of the two power input options.

3. Propeller 8-core Microcontroller System

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

4. 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.

5. Servo Ports

There are three 3 pair of servo port among which one is for ultrasonic range sensor and other two for two wheels. 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. Fig 3.3 shows the different servo parts of the Activity Board.

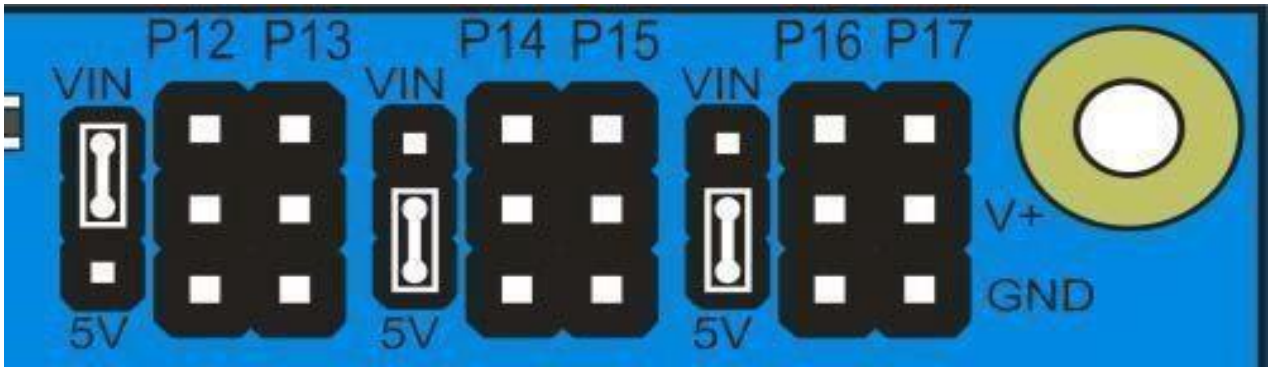


Fig. 3.3 Different Servo Parts

The connections for the servo ports to the servo motors, encoders and range sensors are as follows: P12 - Left Servo

P13 - Right Servo

P14 - Left Encoder

P15 - Right Encoder

P16 - Laser Range Sensor Servo

6. 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.

7. 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".

8. 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.

9. 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.

10. 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.

11. 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.

12. XBee Access

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.

13. Reset Button

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.

14. Power Switch The power switch has 3 settings:

- 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.

15. Stereo Output Jack

The audio output jack fits 1/8" headphone or speaker plugs. Propeller I/O pins P26 (left channel) and P27 (right channel) are hardwired to a low-pass filter, amplifier and coupling capacitor circuits that can drive headphones, ear-buds, speakers with built-in amplifiers, or line level inputs. it is compatible with the Veho 360 speaker from Parallax (item #90000018).

16. USB Port

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.

17. X Bee DO/DI Activity Lights

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.

18. Micro SD Card Socket

This socket is useful for applications that need to access files from a micro SD card.

Examples include:

- Large lookup tables
- Play audio files

- C language programs using the extended memory model (XMM)
- Data logging
- Multiple Spin program application images to be boot loaded

This socket is hardwired to I/O pins: P22 - DO (data out); P23 - CLK (clock); P24 - DI/CD (data in and card detect); P25 - /CS (active low chip select).

Power & Connect Circuits

The 3-Position Power Switch

All versions of the Propeller Activity Board have a 3-position power switch as shown in Fig 3.4

Position 0 — *For building and modifying circuits*. Position 0 turns off power to the circuits on the board.

Position 1 — *For programming and breadboard circuits*. Position 1 connects power to most of the board, including the black sockets along the three edges of the white breadboard. It does NOT connect power to the 3-pin servo ports labeled P12-P17 that are above the white breadboard.

Putting the switch in Position 1 before loading a navigation program that will make the robot's wheels turn, and after that the program will be loaded into EEPROM. This will keep the robot from driving off the table as soon as the program is loaded.

Position 2 — *For making the Activity Bot move*. Position 2 powers all the circuits on the board, including the 3-pin servo ports labeled P12-P17.After loading a navigation program into EEPROM, the robot can be put on the floor, holding the reset button down, switch will be put on position 2. The program will start running.

Fig: 3.4 Power Switch

Blink Lights

Activity Board has two built-in light-emitting diodes (LEDs) near the bottom-right corner of the board. They are already electrically connected to Propeller I/O pins P26 and P27 as shown in Fig 3.5. These LEDs are helpful when developing applications that use sensors.

The idea is to build a program that turns on an LED when a sensor is activated.

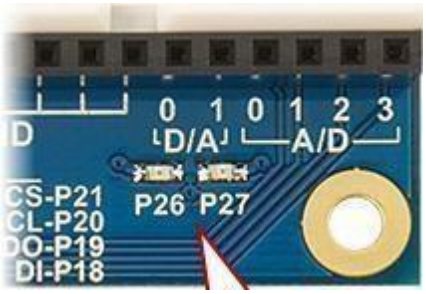


Fig: 3.5 Blink Link

Here are some symptoms and causes.

P26 Light stays off while turning the right wheel.

- The right encoder cable may be plugged into the P14 servo port backwards.
- 20 k resistor (red-black-brown) may not be making contact at either the P14 or 3.3 V socket.

P27 Light stays off while turning the left wheel.

- The left encoder cable may be plugged into the P15 servo port backwards.
- 20 k resistor (red-black-brown) may not be making contact at either the P15 or 3.3 V socket.

P27 light instead of P26 light blinks while wheel turning the right wheel (or vice versa).

- The encoder cables are swapped! Switch the encoder cables plugged into P14 and P15.

P26 or P27 light stays on while turning wheel.

- Resistor connecting P14 or P15 socket to 3.3 V socket is too small. It should be 20 kohm (red-black-orange-gold). This resistor came in the bag with the encoder parts, not with the rest of the resistors in the kit.

3.2.2 Propeller Microcontroller Chip

The multi-core Propeller microcontroller chip is designed to provide high-speed processing for embedded systems while maintaining low current consumption and a small physical footprint. In addition to being fast, the Propeller provides flexibility and power through its eight processors, called cogs, that can perform simultaneous independent or cooperative tasks, all while maintaining a relatively simple architecture that is easy to learn and utilize.

The Fig 3.6 shows the propeller chip along with pins and pin names.

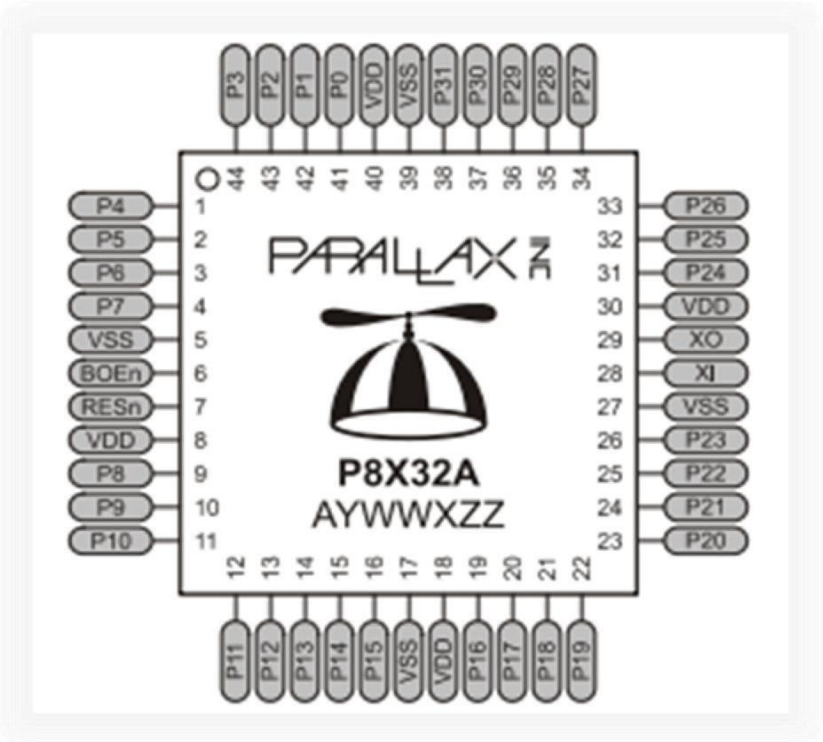


Fig: 3.6 Propeller Chip Pins

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)

3.2.3 High Speed Continuous Rotation Servo Motor

Parallax's high speed continuous rotation servo motor as shown in Fig: 3.7 offers easily controlled bi-directional rotation via simple pulse width modulation. Two such motors are used for rotating the wheels of the mobile robot.

Features

- Bi-directional continuous rotation.
- Up to 150 RPM @ 6 VDC, or 180 RPM at 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.
- 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 at 7.4 VDC when operating in no load conditions, 15 mA at 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) at 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

Fig: 3.7 Parallax High Speed Continuous Rotation Servo Motor

Quick-Start Circuit

Connection of the servo to the Propeller microcontroller is shown in Fig: 3.8.

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

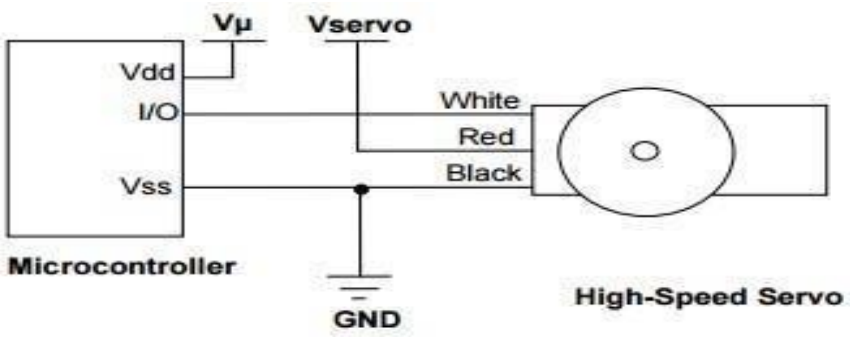


Fig: 3.8 Servo Motor Connection to Propeller Microcontroller

Servo Control

The Parallax high speed continuous rotation servo is controlled through pulse width modulation. Rotational speed and direction are determined by the duration of a high pulse, refreshed every 20 ms. The pulse train required for the servo motor showing the control pulse width for stand still, clockwise and counterclockwise rotation is depicted in Fig: 3.9.

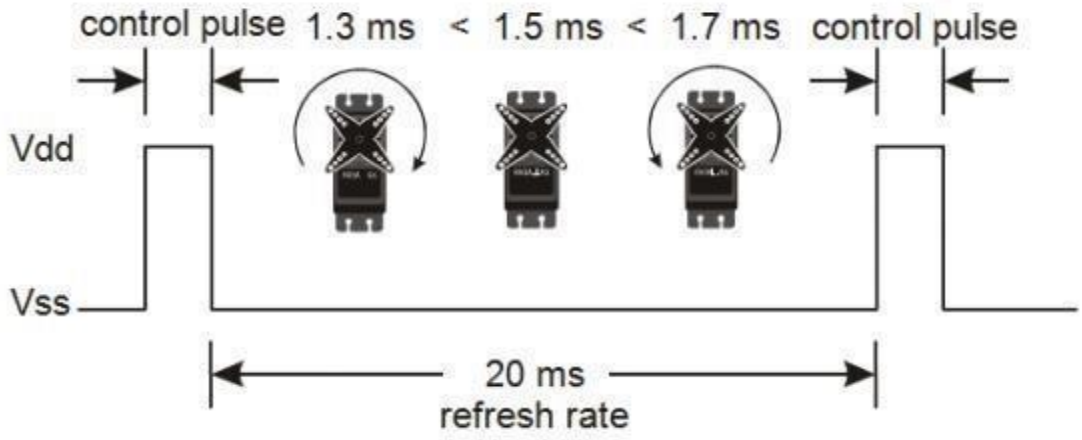


Fig: 3.9 Pulse Train for Servo Motor

- A 1.5 ms control pulse makes the servo stand still.
- As pulse width decreases from 1.5 ms to 1.3 ms, the servo gradually rotates faster, clockwise.
- As pulse width increases from 1.5 ms to 1.7 ms, the servo gradually rotates faster, counterclockwise.

3.2.4 Optical Encoder

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, othere are a total of 64 encoder ticks per wheel turn. A sample of an optical encoder is shown below in Fig: 3.10.

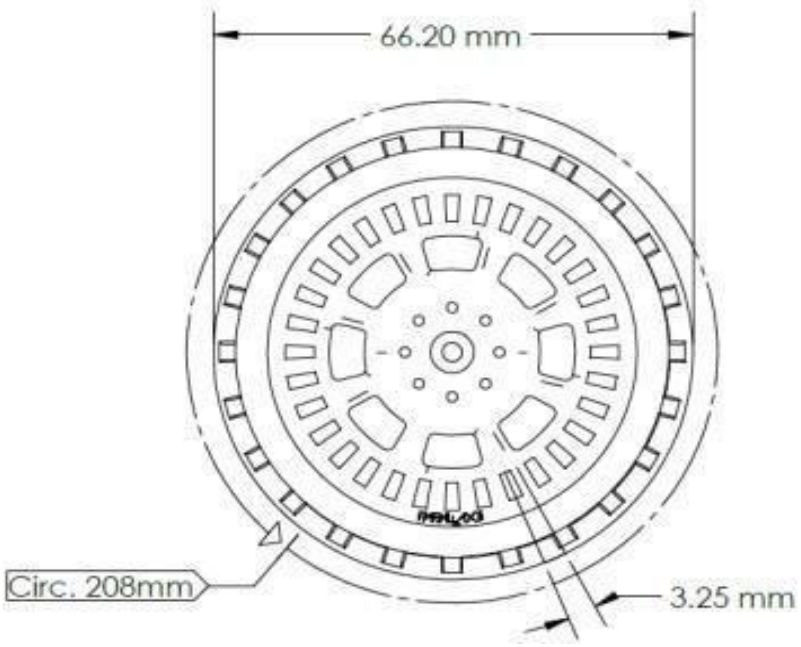


Fig: 3.10 Optical Encoder

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.

3.3 LASER RANGE SENSOR SYSTEM

The LASER range sensor system consists of a LASER transmitter and a LASER receiver for measuring the distance of an object. Proprietary signal processing techniques have been used to achieve high sensitivity, speed, and accuracy in a small, low-power, and low-cost system. The laser range sensor is shown in Fig. 3.11.

Fig. 3.11 LASER Range Sensor

Working Principle of the LASER Range Sensor System

The LASER range sensor system measures distance by calculating the time delay between the transmission of a near-infrared laser signal and its reception after reflecting off from a target, using the known speed of light, as shown in Fig. 3.12. Its 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.

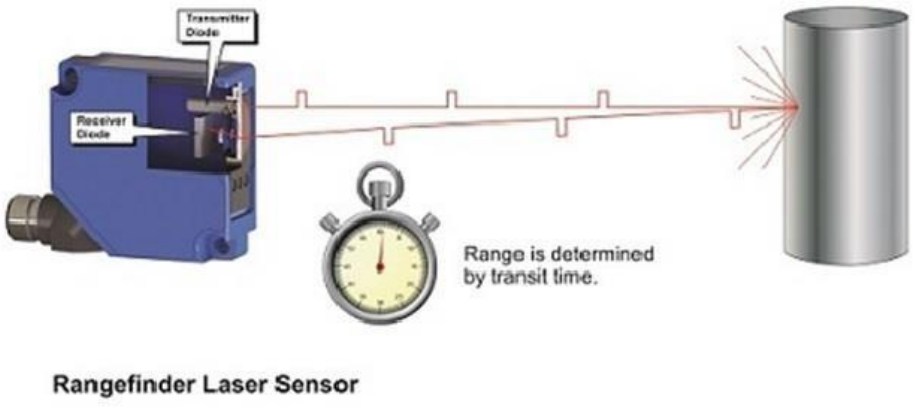


Fig. 3.12 Working Principle of LASER Range Sensor

The device sends a reference signal (signature) 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.

Specifications

Physical

Specification	Measurement
Size (LxWxH)	$20 \times 48 \times 40$ mm$(0.8 \times 1.9 \times 1.6in)$.
Weight	22 g(0.78oz.)
Operating temperature	-20 to 60°C$(-4$ to 140°F$)$

Electrical

Specification	Measurement
Power	5Vdc nominal 4.5Vdc min., 5.5Vdc max.

Current consumption	105 mA idle 135 mA continuous operation

Performance

Specification	Measurement
Range (70% reflective target)	40 m(131ft)
Resolution	$+/-1$ cm(0.4in.)
Accuracy $<$ 5 m	$\pm$2.5 cm (1 in.) typical *
Accuracy $\geq$ 5 m	$\pm$10 cm (3.9 in.) typical Mean $\pm$1% of distance maximum Ripple $\pm$1% of distance maximum
Update rate (70% Reflective Target)	270 Hz typical 650 Hz fast mode* $>$ 1000 Hz short range only
Repetition rate	$\sim$ 50 Hz default 500 Hz max

*Nonlinearity present below 1 m (39.4 in.)

** Reduced sensitivity

Interface

Specification	Measurement
User interface	I2C PWM External trigger
I2C interface	Fast-mode (400 kbit/s) Default 7-bit address 0 $\times$ 62 Internal register access & control
PWM interface	External trigger input PWM output proportional to distance at $10\mu s/\text{cm}$

Laser

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

Connections

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

3.4. ACTIVITYBOT SYSTEM SOFTWARE

The Propeller microcontroller is run by Simple IDE software, and the programming language used for controlling the ActivityBot mobile robot system is Propeller C.

3.4.1 Simple IDE

Simple IDE software is available for Windows and Mac. This is an open-source C programming environment for the multi-core Propeller microcontroller. Simple IDE supports the C, C++, and Propeller Assembly (PASM) programming languages. It comes packaged with the PropGCC compiler for C and C++ and the Open Spin compiler for Propeller Assembly (PASM). Simple IDE has a central workspace to edit code and an integrated serial terminal for exchanging information between user and Propeller microcontroller. Simple IDE uses the Project Manager to keep a list of any library, folder, and file resources the project utilizes that are not part of Propeller GCC. Libraries that are not part of Propeller GCC, or Simple Tools, have to be added to the project so that the compiler can find them. This software helps in writing the program in C language in Propeller C. This is a C-language tutorial for the 8-core Propeller microcontroller. It features the Propeller Activity Board but other Propeller development boards will work. 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.

3.4.2 Propeller C

The version of C language for the Propeller microcontroller is Propeller C.

3.4.3 Main Commands and Functions used in the Project

servo angle

Parallax Standard Servo angle can be set from 0 to 180 degree in tenths of a degree. 0 to 1800 corresponds to control pulses ranging from 500 to 2300 with 1400 at center (90 degrees).

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

pause

Pause causes moving 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.

Example: Pause (int time)

drive_setMax speed

This function sets the maximum speed of the servo motors of the wheels of the mobile robot. 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.

Example: drive_setMax speed (int speed)

drive_goto

This function is used to drive the wheels through a specified distance in either direction. 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.

Example: drive_goto (int distleft, int disright)

sin, cos, atan2

Returns the sine of a radian angle, returns the cosine of a radian angle, returns the arc tangent in radians of y/x based on the signs of both values to determine the correct quadrant respectively.

Example: sin (angle in radian), cos (angle in radian), atan2(y,x).

laser_cm

This command gives measured distance of the nearest obstacle sensed by the LASER range sensor in centimeter.

Example: laser_cm ()

3.5 ACTIVITYBOT NAVIGATION SYSTEM

This section will cover how to move forward, how to move backward, how to rotate left, how to rotate right etc which depend upon direction of rotation of two high speed parallax servo motors. Each wheel is servo controlled and fitted with encoders to calculate the number of ticks it moved. Rotational speed and direction are determined by the duration of a high pulse.

- All commands have to give in ticks for each wheel. Positive ticks number will rotate the wheel in such a direction so as to move the robot in forward directions and negative ticks for backward direction.
- The propeller Activity Bot wheel has 32 spokes, which are separated by 32 spaces, for a total of 64 ticks. If 1/64th of a turn, it will travel 3.25mm.
- If distance is known and if distance is divided by 3.25 then we can find out the ticks value to reach that particular distance.

3.5.1 Forward and Backward Movement

In forward movement the ActivityBot has to turn its both left wheel and right wheel in such a direction so that the robot moves forward, where its left wheel driven by a servo motor connected to pin 12 and right wheel connected to pin 13. The following command is used for forward movment.

```
{drive_goto(int disleft, int disright)      //distance value in ticks and also +ve value for
both ticks  pause(int time)                 // time in mili second  }
```

In backward movement the ActivityBot has to turn its both left wheel and right wheel in the opposite direction of forward movement. The following command is used for backward movement.

```
{drive_goto(int disleft, int disright)      //distance value in ticks and also -ve value for
both ticks  pause(int time)                 // time in mili second }
```

The details of straight line navigation of Activity-Bot in forward and backward direction are depicted in Fig. 3.13.

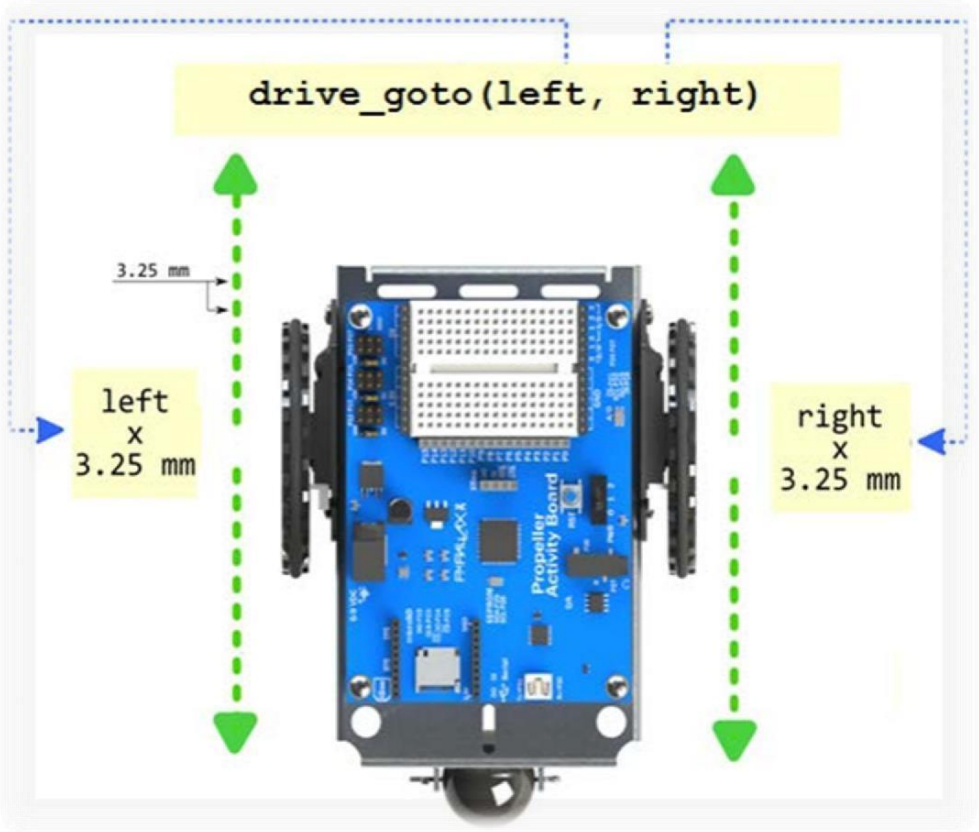


Fig. 3.13 ActivityBot Backward and Forward Movement

3.5.2 Turning the Activity Bot by calculating the encoder ticks

The Propeller ActivityBot wheel has 32 spokes, separated by 32 spaces, for a total of 64 ticks. If the wheel turns 1/64th of a turn, it will travel 3.25 mm. The term "tick" indicates a transition from either spoke detected to hole detected, or vice -versa. The ActivityBot's turning radius(R) is typically 105.8 mm.

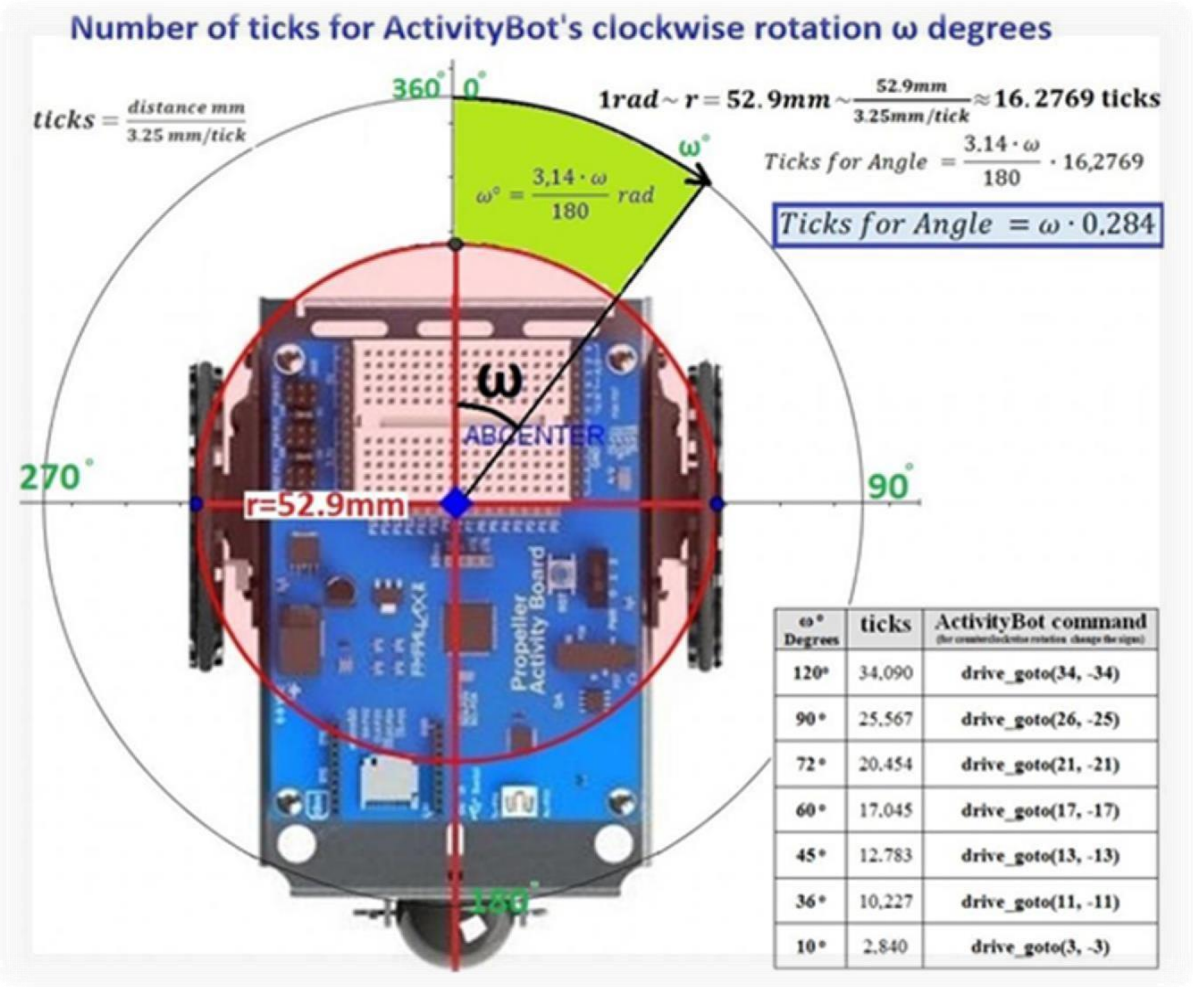
Ticks = distance (mm) ÷ 3.25 mm/tick

Number of turn (n) = turning angle/360

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. 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.

The number of ticks can 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.

For right turning drive_goto (- t/2, t/2) command will be used, for left turning drive_goto (t/2, - t/2) to be used. The Fig. 3.14 shows the basics of turning the Activity-bot by a particular angle by ticks calculation.



ω° Degrees	ticks	ActivityBot command (for counterclockwise rotation change the signs)
120°	34.090	drive_goto(34, -34)
90°	25.567	drive_goto(26, -25)
72°	20.454	drive_goto(21, -21)
60°	17.045	drive_goto(17, -17)
45°	12.783	drive_goto(13, -13)
36°	10,227	drive_goto(11, -11)
10°	2.840	drive_goto(3, -3)

Fig. 3.14 Turning the ActivityBot through a Particular Angle

CHAPTER 4
4.0. EXPERIMENTATION WITH A PARALLAX ACTIVITYBOT MOBILE ROBOT FITTED WITH A LASER RANGE SENSOR FOR NAVIGATION BASED ON IMPROVED E-BUG ALGORITHM

4.1. MODIFICATION IN THE ALGORITHM CONSIDERING THE ACTUAL DIMENSIONS OF MOBILE ROBOT

In E- Bug and all other algorithms 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 obstacle 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 E-Bug algorithm considering the actual dimensions 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 sudden point which is slightly away from the actual sudden point.

4.1.1 Modification in Angle of Rotation for Modified Sudden Point

When the range sensor detects an obstacle, it scans the obstacle on both sides to get sudden points. Now if the mobile robot goes to the actual sudden point based on the selection criteria of E-Bug algorithm, it will definitely collide with the obstacle due its actual size. To avoid the obstacle, the mobile robot, rotate through an extra angle, which should be added to the angle for actual sudden point. The geometry to calculate this extra angle (ang-plus) from the dimension of mobile robot and distance of sudden point, as detected by the LASER range sensor is shown in Fig 4.1 for one direction (right) only. The angle (angr) for the sudden point is obtained with respect to the position of the sensor (point B), but the robot must rotate about the wheel ‘axle’ centre (point A) through an angle, angr + ang- plus, the calculations of which are also shown in the figure.

In the present project, when the mobile robot moves towards goal, it will move up to a distance of 30 cm from any obstacle detected along its path. Considering the distance and other dimensions of the mobile robot (ActivityBot), this extra angle has been calculated as slightly less than 15 °.

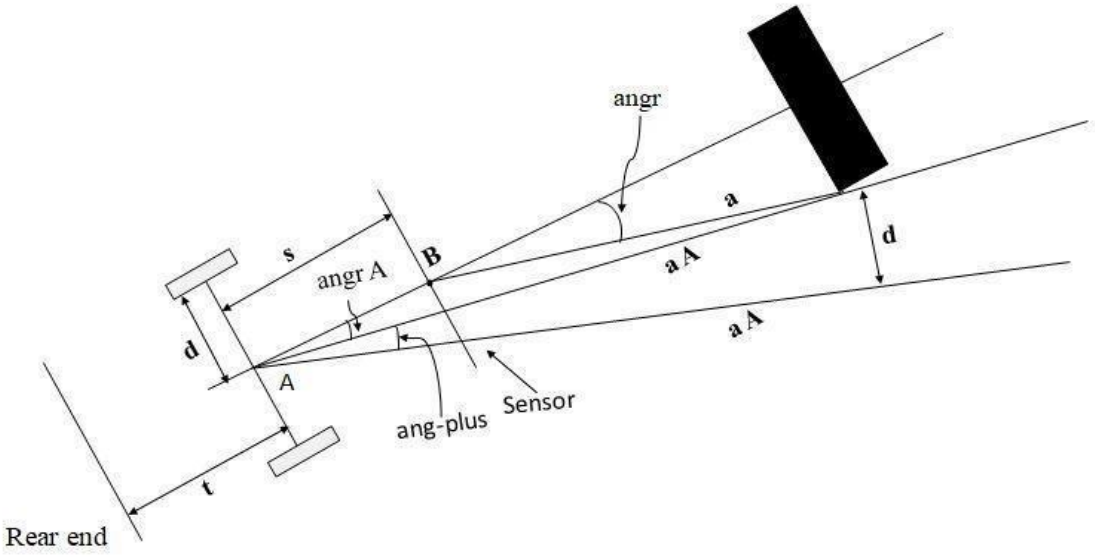


Fig. 4.1 The Geometry for Calculating the Required Modification in Rotation Angle and Distance

B = Location of sensor when the obstacle is detected.

A = Location of robot wheel- 'axle' (imaginary line) center when the obstacle is detected.

angr = Sudden point angle (right side)

angrA = Angle of rotation required for the wheel-axle center.

ang-plus = Minimum addition angle rotation required to avoid collision considering robot width.

a = Sudden point distance from sensor.

a_A = Sudden point distance from wheel- 'axle' center.

d = Distance between wheel- 'axle' center and wheel center (5 cm).

s = Distance between sensor and wheel center (7cm)

t = Distance between the wheel 'axle' center and the rear end of the robot (8.5cm)

Now, ang-plus = d/a_A ……...……. (1)

$a_A . \sin(angr_A) = a.\sin(angr)$……..…. (2)

$a_A . \cos(angr_A) = a.\cos(angr) + s$……..…. (3)

From these equations, a_A, $angr_A$ can be obtained.

4.1.2. Modification in Distance for the Actual Dimension of the Robot

Now after reaching the modified sudden point, the robot will have to reorient itself towards the goal. As the robot will rotate about its wheel axle center to orient itself, the robot should move to a point whose distance will be more than the distance (a) of the sudden point as detected by the range sensor. The geometry to calculate this modified distance is also shown in Fig 4.1. The robot should move to a point so that its end point will reach the modified sudden point and the distance required to move to this point will be $a_A + t$, as shown in the figure where the calculation of a_A is also shown.

4.2 PROGRAM DEVELOPMENT FOR PRODUCING NECESSARY MOVEMENTS OF THE ACTIVITYBOT MOBILE ROBOT USING LASER RANGE SENSOR BASED ON IMPROVED E-BUG ALGORITHM

A program has been developed in Propeller C language for IDE software for producing the necessary movements of the ActivityBot mobile robot using LASER range sensor (capable of 180° rotation) based on the improved E-Bug algorithm. The developed program is used to run the mobile robot from a start to a goal point by avoiding obstacles coming across its path. The coordinates of the start and goal points are entered as variables to execute the program in different layouts of the workspace. The robot is kept parallel to the x-axis at the start position. The Euclidean distance and orientation of goal point from start point is calculated. The robot rotates towards goal and starts moving towards it. The output data for the distance of the nearest object (obstacle) is continuously monitored and the robot moves towards the goal in small steps (taken as 3-ticks = 9.75 mm) until an obstacle is detected within a specified distance (taken as 30 cm) from the range sensor. The coordinates of the current location are calculated using trigonometric relation and formulas of sine and cosine. After detection of an obstacle the range sensor starts searching for the sudden points on both sides by rotating the sensor. The sudden point is taken as a point where the distance measured by the range sensor increases by an amount (taken as 20 cm) such that there is enough space for the mobile robot to go through. The angle of sensor rotation is noted and recorded for determining the angular rotation of the sudden point. But the sensor is further rotated in the same direction to check whether no other obstacle is present within a specified angle (taken as 30° at a distance of 30 cm) such that there is enough space for the ActivityBot mobile robot to go through. If any other obstacle is detected within 30°, the process is repeated and the sudden point is modified, and the corresponding angle of sensor rotation, is also modified. If no sudden point is obtained within 90° of rotation, a message is shown to indicate that it is not possible to move to that direction. Now after the sudden point on both right and left sides are detected, the location of the modified sudden point on both sides are determined considering the actual dimension of the mobile robot and according to the modification done for the angle of rotation and distance as described in sections 4.1.1 and 4.1.2 respectively. These modifications which are necessary for a real mobile robot for its actual dimensions rather than being considered as a point in E-Bug algorithm are the main improvement in the present project in the algorithm for improved E-Bug algorithm. Then the distance from the current point to the modified sudden point and the modified sudden point to target (the latter distance being computed using cosine rule) have been calculated and

added for both sides for finding the total distance to traverse more to reach the goal point (target) for both sides. The robot then ‘chooses’ to move towards the location of the modified sudden point which provides the shorter path. Then the mobile robot rotates towards this modified sudden point by properly calculating the angle of rotation required for this, and moves to this location by properly calculating the distance. After reaching the selected modified sudden point, the robot again rotates towards the target by turning through a properly calculated angle, and again goes towards the goal in steps as it did at the beginning. The whole process is repeated until the robot reaches its goal, or a situation is reached when no path can be found. The simple flowchart of the process is shown in Fig 4.2 and the completed program is given in section 4.3.

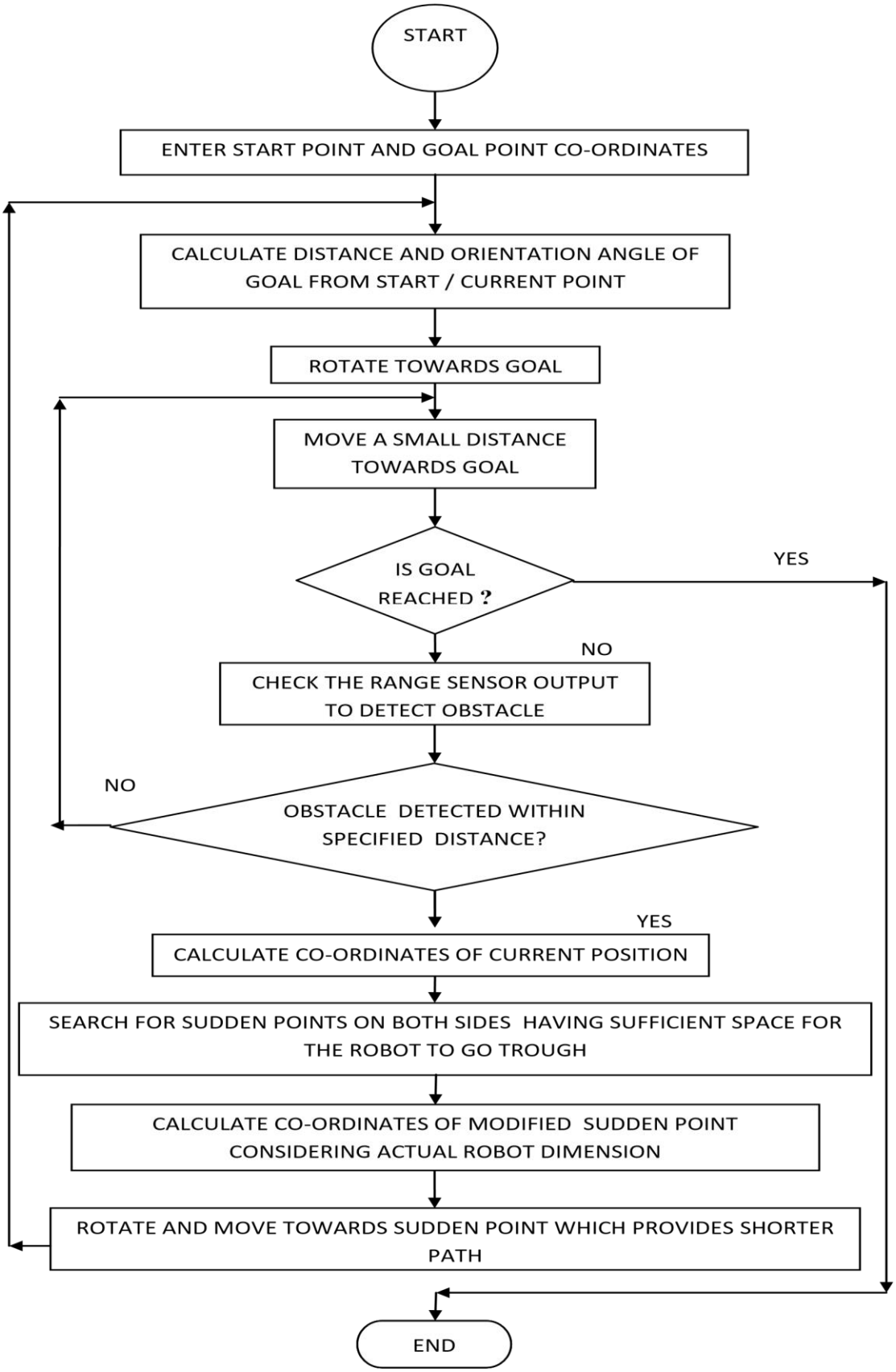


Fig. 4.2 Flowchart of the Developed Program

4.3 PROPELLER C PROGRAM DEVELOPED FOR NAVIGATION OF ACTIVITY-BOT MOBILE ROBOT USING A LASER RANGE SENSOR BASED ON IMPROVEDED E-BUG ALGORITHM

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

```
/*
NAVIGATION OF ACTIVITYBOT MOBILE ROBOT USING A LASER RANGE SENSOR
 BASED ON IMPROVED E-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;
         dr= a+sqrt((a*a)+(d*d)-(2*a*d*cos(angr*pi/180)));
         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;
          dl= b+sqrt((b*b)+(d*d)-(2*b*d*cos(angl*pi/180)));
          print("ang2=%f\n",angl);
          print("b=%f\n",b);

 if(dr<dl)
          {                   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 w.r.t current orientation in angle.

angrad: goal orientation angle w.r.t 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 Xaxis.

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.

4.4. RESULTS AND DISCUSSSIONS

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 range 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 (taken as 30 cm). 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 distance and angle as retrieved from the sensor are modified, as described in sections 4.2 and 4.3. 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. CONCLUTIONS AND FUTURE SCOPE

5.1. CONCLUSIONS

Based on the foregoing analysis, program development, experimentations and results on

"ONLINE PATH PLANNING OF A MOBILE ROBOT USING LASER RANGE SENSOR BASED ON AN IMPROVED E-BUG ALGORITHM" for the present project, the following conclusions may be drawn

a. Various algorithms of different path planning techniques have been studied thoroughly including Bug group of algorithms, where Point Bug and modified Bug algorithms have been found suitable in most cases.

b. Point Bug algorithm has some limitations in finding optimal path because minimizing the angular deviation does not always guarantee shortest path. Some improved algorithms have also been studied which overcomes the deficiencies of Point Bug algorithm.

c. A new sensor based algorithm called E-Bug algorithm is used in the present project. This algorithm is an improved algorithm as compared to other Bug algorithms and is based on the Point Bug algorithm. E-Bug algorithm is simplified E-Bug algorithm finds the near optimal path by adding the sub-paths length from current point to sudden point and from sudden point to target point and selecting the shorter sub-path. The path length from sudden to target point is determined using cosine rule. The sensor output helps in detecting any obstacle present along the path of the robot while travelling the distance in a straight line.

d. Most of the existing path planning algorithms consider mobile robot as a point source and simulations results are available and a very few real robot experimentation of the algorithm is performed. In the present project robot dimension is taken into account while developing the program to prove the effectiveness of the algorithm (E-Bug) in real life situation. Here some modifications have been made in the algorithm for the 'Sudden point'.

e. An arrangement has been made for setting up the workspace consisting of an ActivityBot mobile robot and multiple obstacles.

f. The LASER range sensor has been mounted on the 180 degree rotating servo motor and it is used for scanning both ends of the obstacle for sudden point detection.

g. A program in Propeller C, based on improved E-Bug algorithm for navigating the robot from start to goal position in presence of static obstacles, considering robot dimensions, has been developed using Simple IDE software.

h. The developed program has been run successfully for different workspace layouts and thereby proving the usefulness of the modified E-Bug path planning algorithm in industrial application.

5.2. FUTURE SCOPE

Future scope of the present work includes experimentation of the algorithm in presence of dynamic (moving) obstacles by modifying the algorithm and developing the program accordingly.

CHAPTER 6
6.0. REFERENCE

[1] Sariff, N., & Buniyamin, N. (2006). An overview of autonomous mobile robot path planning algorithms. In *Research and Development,4th Student Conference,*183-188

[2] Alajlan, A., Elleithy, K., Almasri, M., & Sobh, T. (2017). An Optimal and Energy Efficient Multi-Sensor Collision-Free Path Planning Algorithm for a Mobile Robot in Dynamic Environments. *Robotics*, *6*(2), 7-11.

[3] Ng, J., & Bräunl, T. (2007). Performance comparison of bug navigation algorithms. *Journal of Intelligent and Robotic Systems*, *50*(1), 73-84.

[4] Belkhous, S., Azzouz, A., Saad, M., Nerguizian, C., & Nerguizian, V. (2005). A novel approach for mobile robot navigation with dynamic obstacles avoidance. *Journal of Intelligent and Robotic Systems*, *44*(3), 187-201.

[5] Zi-Xing, C. A. I., Zhi-Qiang, W. E. N., ZOU, X. B., & CHEN, B. F. (2008). A Mobile Robot Path-planning Approach under Unknown Environments. *IFAC Proceedings Volumes*, *41*(2), 5389-5392.

[6] Langer, R. A., Coelho, L. S., & Oliveira, G. H. (2007). K-Bug, a new bug approach for mobile robot's path planning. In *Control Applications, IEEE International Conference,* 403-408.

[7] Roy, N., Chattopadhay, R., Mukherjee, A., & Bhuiya, A. (2017). Implementation of Image Processing and Reinforcement Learning in Path Planning of Mobile Robots. *International Journal of Engineering Science,* 7(10) *15211-15213*.

[8] Nguyen, H. T., & Le, H. X. (2016). Path planning and Obstacle avoidance approaches for Mobile robot, International Journal of Computer Science Issues, 13(4), 1-10.

[9] Ganeshmurthy, M. S., & Suresh, G. R. (2015) Path planning algorithm for autonomous mobile robot in dynamic environment. In *Signal Processing, Communication and Networking (ICSCN), 3rd International Conference,* 1-6.

[10] Alpaslan YU, Osman PA (2009). Performance Comparison of Bug Algorithms for Mobile Robots. In 5th International Advanced Technologies Symposium, 13(1).

[11] Devi, B. M., & Prabakar, S. (2013). Dynamic Point Bug Algorithm for Robot Navigation. *International Journal of Scientific & Engineering Research*, *4*(4), 12761279.

[12] Al-Jarrah, R., Al-Jarrah, M., & Roth, H. (2018). A Novel Edge Detection Algorithm for Mobile Robot Path Planning. *Journal of Robotics*.

[13] Han, J., & Seo, Y. (2017). Mobile robot path planning with surrounding point set and path improvement. *Applied Soft Computing*, *57*, 35-47.

[14] Zohaib, M., Iqbal, J., & Pasha, S. M. (2018). A Novel Goal-Oriented Strategy for Mobile Robot Navigation without Sub-Goals Constraint. *Revue Roumaine Des Sciences TechniquesSerie Electrotechnique Et Energetique*, *63*(1), 106-111.

[15] Borenstein, J., & Koren, Y. (1989). Real-time obstacle avoidance for fast mobile robots. *IEEE Transactions on systems, Man, and Cybernetics*, *19*(5), 1179-1187.

[16] Harshini, K., &, Ramji. (2015). Navigation of mobile robots in the presence of static obstacles of various shapes. *5*(10-11),

[17] Sankaranarayanan, A., & Vidyasagar, M. (1990). A new path planning algorithm for moving a point object amidst unknown obstacles in a plane. In *Robotics and Automation, Proceedings, International Conference*, 1930-1936.

[18] Buniyamin, N., Ngah, W. W., Sariff, N., & Mohamad, Z. (2011). A simple local path planning algorithm for autonomous mobile robots. *International journal of systems applications, Engineering & development*, *5*(2), 151-159.

[19] Mandal, P., Barai, R. K., Maitra, M., & Roy, S. (2013). Path planning of autonomous mobile robot: A new approach. In *Intelligent Systems and Control (ISCO), 7th International Conference*, 238-243

[20] Hachour, O. (2008). Path planning of Autonomous Mobile robot. *International journal of systems applications, engineering & development*, *2*(4), 178-190.

[21] Iijima, J. I. (1992). Searching unknown 2-D environment by a mobile robot with a range sensor. *Computers & electrical engineering*, *18*(1), 83-98.

[22] Contreras-Cruz, M. A., Ayala-Ramirez, V., & Hernandez-Belmonte, U. H. (2015). Mobile robot path planning using artificial bee colony and evolutionary programming. *Applied Soft Computing*, *30(1)*, 319-328.

[23] Kuric, I., Bulej, V., Saga, M., & Pokorny, P. (2017). Development of simulation software for mobile robot path planning within multilayer map system based on metric and topological maps. *International Journal of Advanced Robotic Systems*, *14*(6), 1-14.

[24] Cai, L., Yang, J., Zhao, L., & Wu, L. (2018). An efficient optimization algorithm for quadratic programming problem and its applications to mobile robot path planning. *International Journal of Advanced Robotic Systems*, *15*(1), 1-8.

[25] Jin, T., Ko, J., & Lee, J. (2004). An efficient path planning of a mobile robot using image of a moving object. *IFAC Proceedings Volumes*, *37*(12), 291-296.]

[26] Pandey, A., & Parhi, D. R. (2017). Optimum path planning of mobile robot in unknown static and dynamic environments using Fuzzy-Wind Driven Optimization algorithm. *Defence Technology*, *13*(1), 47-58.

[27] Leena, N., & Saju, K. K. (2014). A survey on path planning techniques for autonomous mobile robots. *IOSR Journal of Mechanical and Civil Engineering (IOSR-JMCE)*, *8*, 7679.

[28] Sahu, D., & Mishra, A. K. (2017). Mobile Robot Path Planning by Genetic Algorithm with Safety Parameter. International Journal of Engineering Science, 14723.

[29] Akka, K., & Khaber(2018). Mobile robot path planning using an improved ant colony optimization. International Journal of Advanced Robotic.

[30] Haj Darwish, A., Joukhadar, A., & Kashkash, M. (2018). Using the bees algorithm for wheeled mobile robot path planning in an indoor dynamic environment. Cogent Engineering, (just-accepted), 1426539.

[31] Li, G., & Chou, W. (2018). Path planning for mobile robot using self-adaptive learning particle swarm optimization. Science China Information Sciences, 61(5), 052204.

[32] Kumar, E. V., Aneja, M., & Deodhare, D. (2017). Solving a Path Planning Problem in a Partially Known Environment using a Swarm Algorithm. arXiv preprint arXiv:1705.03176.

[33] Khatib, O. (1986). Real-time obstacle avoidance for manipulators and mobile robots. In *Autonomous robot vehicles* (pp. 396-404). Springer, New York, NY.

[34] Sedighi, K. H., Ashenayi, K., Manikas, T. W., Wainwright, R. L., & Tai, H. M. (2004). Autonomous local path planning for a mobile robot using a genetic algorithm. In *Evolutionary* Computation, *CEC, Congress*, 2(1), 1338-1345

[35] Lawler EL (1976) combinatorial optimization. Holt, Rinehart and Winston, New York

[36] Lumelsky, V. J., & Stepanov, A. A. (1987). Path-planning strategies for a point mobile automaton moving amidst unknown obstacles of arbitrary shape. *Algorithmica*, *2*(1-4), 403-430.

[37] Meddah, F., & Dib, L. (2015). E-Bug: New Bug Path-planning algorithm for autonomous robot in unknown environment. In *Proceedings of the International Conference on Intelligent Information Processing, Security and Advanced Communication*, 1-8, DOI:10.1145/2816839.2816864

[38] Banerjee, S. (2014). A comparative study of underwater robot path planning algorithms for adaptive sampling in a network of sensors, 1-159

[39] Kamon, I., & Rivlin, E. (1997). Sensory-based motion planning with global proofs. *IEEE transactions on Robotics and Automation*, *13*(6), 814-822.

[40] Noborio, H., Nogami, R., & Hirao, S. (2004). A new sensor-based path-planning algorithm whose path length is shorter on the average. In *Robotics and Automation, Proceedings, IEEE International Conference*, 3(1), 2832-2839

[41] Langer, R. A., Coelho, L. S., & Oliveira, G. H. (2007). K-Bug, a new bug approach for mobile robot's path planning. In *Control Applications, IEEE International Conference* 403-408, **DOI:** 10.1109/CCA.2007.4389264

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