ONLINE PATH PLANNING OF A MOBILE ROBOT WITH AN ULTRASONIC RANGE SENSOR USING E-BUG ALGORITHM

By

BASIR AHMED

B.TECH. (Mechanical Engineering), 2013

KALYANI GOVERNMENT ENGINEERING COLLEGE

EXAMINATION ROLL No.: M4PRD1608

REGISTRATION No.: 129417 of 2014-15

THESIS

SUBMITTED IN PARTIAL FULLFILLMENT OF THE REQUIREMENTS FOR THE AWARD OF THE DEGREE OF MASTER OF PRODUCTION ENGINEERING IN THE FACULTY OF ENGINEERING AND TECHNOLOGY,

JADAVPUR UNIVERSITY, 2016

DEPARTMENT OF PRODUCTION ENGINEERING

JADAVPUR UNIVERSITY

KOLKATA-700032

JADAVPUR UNIVERSITY

FACULTY OF ENGINEERING AND TECHNOLOGY

CERTIFICATE OF RECOMMENDATION

DATE:_____

I HEREBY RECOMMENDED THAT THE THESIS PREPARED UNDER MY SUPERVISSION BY **BASIR AHMED** ENTITLED **"ONLINE PATH PLANNING OF A MOBILE ROBOT WITH AN ULTRASONIC RANGE SENSOR USING E-BUG ALGORITHM"** BE ACCEPTED IN PARTIAL FULLFILLMENT OF THE REQUIREMENT FOR THE DEGREE OF **"MASTER OF PRODUCTION ENGINEERING".**

COUNTERSIGNED

_____

THESIS ADVISOR

(Prof. Ajoy Kumar Dutta)
Production Engineering Dept.
JADAVPUR UNIVERSITY
KOLKATA- 700032

_____

Dr. Debamalya Banerjee
HEAD OF THE DEPARTMENT
Production Engineering Department
JADAVPUR UNIVERSITY
KOLKATA-700032

_____

Dr. Sivaji Bandyopadhyay
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 satisfactory to warrant its acceptance as a prerequisite to the degree for which it has been submitted. It is understood that by this approval the undersigned do not necessarily endorse or approve any statement made, opinion expressed or conclusion drawn therein, but approve the thesis only for the purpose for which it is submitted.

COMMITEE ON _____

FINAL EXAMINATION _____

FOR EVALUATION OF _____

THE THESIS _____

*Only in case the recommendation is concurred in

DECLARATION OF ORIGINALITY AND COMPLIANCE OF ACADEMIC ETHICS

I hereby declare that this thesis contains literature survey and original research work by the undersigned candidate, as part of his Master of Production Engineering studies.

All information in this document have been obtained and presented in accordance with academic rules and ethical conduct.

I also declare that, as required by these rules and conduct, I have fully cited and referenced all materials and results that are not original to this work.

Name : BASIR AHMED

Examination Roll Number: M4PRD1608

Class Roll Number : 001411702013

Thesis Title: **ONLINE PATH PLANNING OF A MOBILE ROBOT WITH AN ULTRASONIC RANGE SENSOR USING E-BUG ALGORITHM.**

_____

Signature with date

ACKNOWLEDGEMENTS

I owe a deep debt of gratitude to my thesis advisor **Prof. Ajoy Kumar Dutta**, Department of Production Engineering, Jadavpur University for his invaluable and untiring guidance, encouragement and supervision, throughout this research work. His effective skill knowledge and experience have made it possible for me to successfully complete the thesis work within the stipulated time. I am very much indebted to him and express my sincere gratitude to him.

Special thanks to **Mr. Subir Kumar Debnath**, Associate Professor, Department of Production Engineering, Jadavpur University, for his encouragement and kind support.

I also convey my deepest regards to **Dr. Debamalya Banerjee**, Head of Production Engineering Department for his kind support.

I also take this opportunity to express my gratitude to all the faculty members of Production Engineering Department for their mental support, immense help and co-operation during the course of this thesis work.

I express my heartiest thanks to my friends and classmates and all those people, who some time or the other directly or indirectly rendered their help at different stages of this work.

I am also thankful to the librarian and research scholars of Production Engineering Department, Jadavpur University for their cordial assistance.

I am ever indebted to my parents for their blessings in each synergy of life. Everything in this nature is time bound, so thanks to 'Almighty' for successful completion of the work in time.

Date:_____ _____

BASIR AHMED

TABLE OF CONTENTS

Chapter- 1

INTRODUCTION

1.1 INTRODUCTION TO MOBILE ROBOT

The term “ROBOT” comes from a Czech word, “ROBOTA” meaning “forced labor”. The word robot first appeared in a 1920 play, by Czech writer Karel Capek, R.U.R: Rossum’s universal robots. The first commercial robot was developed in 1961 and used in the automotive industry by Ford. First-generation robots date from the 1970s and consist of stationary, nonprogrammable, electromechanical devices without sensors. Second-generation robots were developed in 1980s and were able to contain sensors and programmable controllers. Third-generation robots were developed between approximately 1990 and the present. In the 1950s, the earliest robots were able to achieve basic locomotion. In the 1970s and 80s, the optimal completion of global objectives dominated the active research. Later, real-time constraints gained attention at the cost of optimal planning [1].

The revolution in the field of mechatronics has made it possible to see the ‘fiction’ robots in reality in various fields of life ranging from articulated robots to mobile robots. Mobile robots permit human access to unreachable locations including accidental situations like fire, building collapse, earthquake and hazardous scenarios such as Nuclear Power Plant (NPP), chemical industry, transmission lines etc. Deployment of a mobile robot in real world applications demands addressing several new issues regarding robot interaction. The increasing development in robotics has brought up various challenges including obstacle avoidance, path planning, navigation, localization, autonomous control etc. Intelligence in the robot navigation to achieve autonomy is a challenging problem for researchers, as it is an important task to design the robot which can perform variety of tasks, such as surveillance, transportation, exploration or human locomotion. In the field of mobile robotics, intelligent obstacle avoidance is the most important task, since every autonomous robot has to plan a safe path for its trajectory towards destination. This is achieved with an intelligent algorithm that uses knowledge of goal position and the sensorial information of the surrounding environment.

1.2 PATH PLANNING OF MOBILE ROBOT

Path planning for mobile robots is one of the most important aspects in robot navigation [2]. The main goal of the robot path planning is to search a safe path for a mobile robot, to make the robot move from the start point to the destination point without collision with obstacles. Also, the path is often required to be optimal in order to reduce energy consumption and communication delay.

The general problem of path planning for mobile robots is defined as the search for a path which a robot (with specified geometry) has to follow in a described environment, in order to reach a particular position and orientation, given an initial position and orientation. As mobile robot is not a point in space, it has to determine the correct direction or perform a proper movement to reach destination and this is called maneuvering planning. Path planning is the determination of a path that a robot must take in order to pass over each point in an environment and path is a plan of geometric locus of the points in a given space where the robot has to pass through. Generally, the problem of path planning is about finding paths by connecting different locations in an environment such as graph, maze and road. Path planning "enables" mobile robots to see the obstacle and generate an optimum path so as to avoid them.

Every path planning method is dependent on a state space. State space represents all possible positions and orientations of a robot. Usually stat space is represented implicitly by a planning algorithm. That is why the path planning algorithms can be divided on the basis of used state space. The mostly used representation of state space in robotics is grid/metric map. Although geometric or topological representations are used, the metric map is easy to update new information about the environment.

The existing robot navigation algorithms can be categorized into three main types on the basis of knowledge of environment: completely known, partially known and unknown environment. In completely known environment, it is easy to tune a robot, by simply creating a map and to generate a reference path. On the other hand, in a partially or completely unknown environment, an obstacle avoidance algorithm [3] is required. Obstacle avoidance is the backbone of autonomous control in the robot navigation

especially in a fully unknown environment as it plays the key role in safe path planning. The algorithm for this purpose must be efficient enough, so that it can take a quick decision while encountering an obstacle without human intervention. Wide sample space of obstacle shapes that can be encountered in real world applications further necessitates the algorithm intelligence.

In the field of mobile robotics, intelligent obstacle avoidance is the most important task, since every autonomous robot has to plan a safe path for its trajectory towards destination. This is achieved with an intelligent algorithm that uses knowledge of goal position and the sensorial information of the surrounding environment. With a focus on these primary features, the present research proposes an intelligent goal-oriented algorithm for autonomous navigation of mobile robots. The proposed algorithm does outperform the existing approaches. It proves the convergence with relatively short, smooth and safe trajectory.

Various approaches have been introduced to implement path planning for a mobile robot. The approaches are according to environment, type of sensor, robot capabilities etc, and these approaches are gradually toward better performance in term of time, distance, cost and complexity. Al-Taharwa [4] for example, categorized path planning as an optimization problem according to definition that, in a given mobile robot and a description of an environment, plan is needed between start and end point to create a path that should be free of collision and satisfies certain optimization criteria such as shortest path. This definition is correct if the purpose of solving path planning problem is for the shortest path because most new approaches are introduced toward shorter path. Looking for the shorter path does not guarantee the time taken is shorter because sometime the shorter path needs complex algorithm making the calculation to generate output is longer.

1.3 LITERATURE REVIEW

Extensive research works have been carried out in the field of mobile robot path planning in last two decades. Some of those researches are cited below:

Khatib, O [3] presented a unique real-time obstacle avoidance approach for manipulators and mobile robots based on the "artificial potential field" concept. In this approach, collision avoidance, traditionally considered a high level planning problem can be effectively distributed between different levels of control, allowing real-time robot operations in a complex environment. This can be applied obstacle avoidance scheme to robot arm using a new approach to the general problem of real-time manipulator control and reformulated the manipulator control problem as direct control of manipulator motion in operational space, in which the task is originally described-rather than as control of the task's corresponding joint space motion obtained only after geometric and kinematic transformation.

Vladimir J. Lumelsky and Alexander A. Stepanov [5] studied the problem of path planning for the case of a mobile robot moving in an environment filled with obstacles whose shape and positions are not known. Under the accede model, the automaton knows its own and the target coordinates, and has a "sensory" feedback which provides it with local information on its immediate surroundings. This information is shown to be sufficient to guarantee reaching a global objective (the target), while generating reasonable (if not optimal) paths.

Lumelsky,V., T. Skewis [2] proposed a model of mobile robot navigation is considered whereby the robot is a point automaton operating in an environment with unknown obstacles of arbitrary shapes. The robot's input information includes its own and the target point coordinates, as well as local sensing information such as from stereo vision or a range finder. These algorithmic issues are addressed 1) Is it possible to combine sensing and planning functions, thus producing, similar to the way it is done in nature, "active sensing" guided by the needs of planning? (The answer is "yes"). 2) Can richer sensing (e.g., stereo vision versus tactile) guarantee better performance, that is, resulting in shorter paths? (The general answer is "no.") A paradigm for combining range

data with motion planning is presented. It turns out that extensive modifications of simpler “tactile” algorithms are needed to take full advantage of additional sensing capabilities.

Sankaranarayanan, A., M. Vidyasagar [6] proposed a new sensor-based path-planning algorithm running in an uncertain 2D world. In the algorithm, the automaton basically goes straight to the goal, and if it touches some uncertain obstacles, it traces the obstacle boundary until it can go straight to the goal again. Moreover if the automaton finds some loop in the world, it enters into the goal area. As a result, several loops encountered by the automaton become smaller monotonously in the world, and then the automaton arrives at the goal after it has escaped from the minimum loop. This result was ensured in an online manner. The algorithm has several good characteristics: (1) it omits some calculations of the distance from the present point, and therefore it is faster than all previous sensor-based algorithms; (2) it is more robust than the previous sensor-based algorithms for some self-dead-reckoning error; (3) it can generate shorter deadlock-free paths in almost all 2D worlds than the previous sensor-based algorithms.

Kamon, I., E. Rivlin [7] proposed a sensory based algorithm DistBug, that was guaranteed to reach the target in an unknown environment or report that the target was unreachable, was presented. The algorithm is reactive in the sense that it relies on range data to make local decisions, and does not create a world model. The algorithm consists of two behaviors (modes of motion): straight motion between obstacles and obstacle boundary following. Simulation results as well as experiments with a real robot are presented. The condition for leaving obstacle boundary is based on the free range in the direction to the target. This condition allows the robot to leave the obstacle as soon as the local conditions guarantee global convergence. Range data is utilized for choosing the turning direction when the robot approaches an obstacle. A criterion for reversing the boundary following direction when it seems to be the wrong direction is also introduced.

Kamon, I., E. Rimon, E. Rivlin [8] TangentBug is a new algorithm in this family, specifically designed for using a range sensor. TangentBug uses the range data to compute a locally shortest path, based on a novel structure termed the local tangent graph (LTG). The robot uses the LTG for choosing the locally optimal direction while moving toward the target, and for making local short-cuts and testing a leaving condition while moving along an obstacle boundary. The transition between these two modes of motion is governed by a globally convergent criterion, which is based on the distance of the robot from the target and analyze the properties of TangentBug, and presented simulation results that show that Tangent Bug consistently performs better than the classical Bug algorithms.The simulation results also show that TangentBug produces paths that in simple environments approach the globally optimal path, as the sensor's maximal detection-range increases. The algorithm can be readily implemented on a mobile robot, and we discuss one such implementation.

Noborio, H., K. Fhjimura, Y. Horiuchi [9] proposed an unknown maze has few collision-free paths to a destination. Therefore, a robot supervised by the classic sensor-based path-planning algorithms Bug2, Class1, Alg1, Alg2 repeatedly enters into long local and global loops excluding and including a destination (goes out of its true way), respectively. For example, in Alg1 and Alg2, we can point out a case that a robot always enters into a global loop one time, and also in Bug (alter.) and Class1 (alter.), we can find another case that a robot frequently joins a local loop many times. A complicated maze usually includes such cases, and therefore a robot arrives at a destination via a very long collision-free path. To overcome this, we revisit an algorithm, HD-I, whose following direction is adequately changed by trial and error. In HD-I, a robot hardly select an inadequate direction and consequently decreases a probability to enter into global and local loops.

John E. Bella, Patrick R. McMullenb [10] applied the meta-heuristic method of ant colony optimization (ACO) to an established set of vehicle routing problems (VRP). The procedure simulates the decision-making processes of ant colonies as they forage for food and is similar to other adaptive learning and artificial intelligence techniques

such as Tabu Search, Simulated Annealing and Genetic Algorithms. Modifications are made to the ACO algorithm used to solve the traditional traveling salesman problem in order to allow the search of the multiple routes of the VRP.

N. Bin, C.X., Z. Liming, X. Wendong [11] shown a novel model of organized neural network is very effective for path planning and obstacle avoidance in an unknown map which is represented by topologically ordered neurons. With the limited information of neighbor position and distance of the target position, robot will autonomously provide a proper path with free-collision and no redundant exploring in the process of exploring.

Xiao-Guang Gao, Xiao-Wei Fu, Da-Qing Chen [12] presents a genetic-algorithm-based approach to the problem of UAV path planning in dynamic environments. Variable-length chromosomes and their genes have been used for encoding the problem. We model the vehicle path as a sequence of speed and heading transitions occurring at discrete times, and this model specifically contains the vehicle dynamic constraints in the generation of trial solutions. Simulation studies have shown that the proposed algorithm is effective in finding a near-optimal obstacle-free path in a dynamically changing environment, and the algorithm can guarantee that all candidate solutions lie within a feasible and reachable path space.

Chestnutt, Lau and Cheung used a modified A* algorithm [13] to calculate path for a Honda ASIMO humanoid robot. The path planning method is applied to real robots rather than simulated on software. A grid of cell is employed to represent the environment. Colour cell represents the obstacles. The cells create a bitmap representing the free spaces and obstacles in the map. The algorithm plans a sequence of footstep positions to navigate toward a goal location based on known static and moving obstacles with predictable trajectories. It uses three cost functions to constrain the step nodes used by the robot.

(1)The first cost is the location cost, which evaluates the foot step location with regard to the environment to determine if the location is safe place to step.
(2) The second cost is the step cost, which calculates the cost to the robot to make the

desired step.

(3) The last cost is the estimated remaining cost-to-go, which is calculated by planning backwards from the goal with a standard mobile robot planner as a pre-computation step. This cost is used to avoid to the local minimums.

Santiago Garrido et al. [14] presented a new sensor based global path planner which operates in two steps. In the first step the safest areas in the environment are extracted by means of a Voronoi diagram. In the second step fast marching method is applied to the Voronoi extracted areas in order to obtain the shortest path. In this way the trajectory obtained is the shortest between the safe possible ones. This two step method combines an extremely fast global planner operating on a simple sensor based environment modeling, while it operates at the sensor frequency. The main characteristics are speed and reliability, because the map dimensions are reduced to a uni-dimensional map and this map represents the safest areas in the environment for moving the robot.

Ng, J., T. Braunl [15] proposed eleven variations of Bug algorithm have been implemented and compared against each other on the EyeSim simulation platform and discusses their relative performance for a number of different environment types as well as practical implementation issues.

Wang and Sillitoe proposed a vertices genetic algorithm [16] planner in 2007.The planner is able to rapidly determine optimal or near-optimal solutions for a mobile robot in an environment with moving obstacles. The method uses the vertices of the obstacles as search space and produces off-line path planning through the environment with dynamic obstacles. It firstly incorporates the robot speed into the genetic genes, which could optimize both the travel time and distance of the robot. Before the robot starts movements, the complete motion knowledge of the moving obstacles in the observed region is available for the robot. The robot uses the genetic algorithm based planner to complete the time or distance-optimized solutions and then starts to travel.

The assumptions of this method are:

1. Obstacles are bounding polygons with vertices.
2. The speed of the moving obstacles is fixed value, and
3. Physical dimensions of the robot are neglected and regarded as a single point.

Jasmin Velagic, et.al proposed a new reactive planning algorithm [17] for mobile robot navigation in unknown environments. The overall navigation system consists of three navigation subsystems. The lower level subsystem deals with the control of the linear and angular velocities using a multivariable PI controller described with a full matrix. The position control of the mobile robot is in the medium level, and it is a nonlinear. The nonlinear control design is implemented by a back stepping algorithm whose parameters are adjusted by a genetic algorithm. The high level subsystem uses the Fuzzy logic and Dempster-Shafer evidence theory to design the fusion of sensor data, map building and path planning tasks. The path planning algorithm is based on a modified potential field method. In this algorithm, the fuzzy rules for selecting the relevant obstacles for robot motion are introduced. Also, suitable steps are taken to pull the robot out of the local minima. A particular attention is paid to detection of the robot's trapped state and its avoidance.

Ismail AL-Taharwa, Alaa Sheta and Mohammed Al-Weshah [18] presented their initial idea for using genetic algorithms to help a controllable mobile robot to find an optimal path between a starting and ending point in a grid environment. The mobile robot has to find the optimal path which reduces the number of steps to be taken between the starting point and the target ending point. GAs can overcome many problems encountered by traditional search techniques such as the gradient based methods. The proposed controlling algorithm allows four-neighbor movements, so that path-planning can adapt with complicated search spaces with low complexities. The results are promising.

Priyadarshi Bhattacharya and Marina L. Gavrilova provided an algorithm based on Voronoi diagram [19] to compute an optimal path between source and destination in the presence of simple disjoint polygonal obstacles. They evaluate the quality of the path based on clearance from obstacles, overall length and smoothness and

provided a detailed description of the algorithm for Voronoi diagram maintenance and dynamic updates. The advantage of the proposed technique versus alternative path-planning methods is in its simplicity, versatility, and efficiency.

M. Tarokh [20] has developed an intelligent path planning approach for highly mobile robots operating in rough environments. The approach consists of characterization of the environment using fuzzy logic, and two-stage GA planners with one being global and the other being local. The global planner determines the path that optimizes a combination of terrain roughness and path curvature; while the local planner uses sensory information, and when previously unknown and unaccounted obstacles are detected, performs on-line re-planning to get around the newly discovered obstacles.

Hu Gu developed an improved algorithm to solve the problem of optimal route planning in vehicle navigation systems [21]. It is based on the standard GA and the lambda-interchange local search method. It can find the optimum route efficiently without any network constraint conditions and can work well in either continuous or discrete networks.

Kumar, E.V., M. Aneja, D. Deodhare [22] proposed the concept of swarm intelligence is based on the collective social behavior of decentralized body, either natural or artificial like ant, fish, bird, bee etc. In this paper presented a review of 4 different algorithms based on swarm intelligence for finding the path by mobile robot. Path planning is an interesting problem in mobile robotics. It is about finding the shortest, collision free and smooth path by the robot form predefined starting position to fixed goal position in an environment with obstacles either moving or stationary.

Santiago Garrido, Luis Moreno, M. Abderrahim and D. Blanco [23] proposed to navigate in complex environments, a robot needs to reach a compromise between the need for having efficient and optimized trajectories and the need for reacting to unexpected vents. This paper presents a new sensor based non-holonomic Path Planner which integrates the global motion planning and local obstacle avoidance capabilities. In the first step the safest areas in the environment are extracted by means of a tube skeleton similar to a Voronoi diagram but with tubular shape. In the second step

Fast Marching Method is applied to the tube skeleton extracted areas in order to obtain the best path in terms of smoothness and safety. At the same time, during the motion, the algorithm modifies the calculated path when it encounters obstacles or other unforeseen dynamic objects that cannot be included in the a priori map. In this way the trajectory obtained is the shortest between the safe possible ones. The method combines map-based and sensor-based planning operations to provide a reliable motion plan, while it operates at the sensor frequency. The main characteristics are speed and reliability, because the map dimensions are reduced to a uni-dimensional map and this map represents the safest areas in the environment for moving the robot.

Ioan Susnea, Viorel Minzu, Grigore Vasiliu [24] proposed a novel, reactive algorithm for real time obstacle avoidance, compatible with low cost sonar or infrared sensors, fast enough to be implemented on embedded microcontrollers. This algorithm "the bubble rebound algorithm". According to this algorithm, only the obstacles detected within an area called "sensitivity bubble" around the robot are considered. The shape and size of the sensitivity bubble are dynamically adjusted, depending on the kinematics of the robot. Upon detection of an obstacle, the robot "rebounds" in a direction having the lowest density of obstacles, and continues its motion in this direction until the goal becomes visible, or a new obstacle is encountered. The performances and drawbacks of the method are described, based on the experimental results with simulators and real robots.

Soh Chin Yun, S. Parasuraman developed a dynamic path planning algorithm [25] in mobile robot navigation. Mobile Robot Navigation is an advanced technique where static, dynamic, known and unknown environment is involved. In this research, Genetic Algorithm (GA) is used to assist mobile robot to move, identify the obstacles in the environment, learn the environment and reach the desired goal in an unknown and unrecognized environment. This study is focused on exploring the algorithm that avoids acute obstacles in the environment. In the event of mobile robot encountering any dynamic obstacles when travelling from the starting position to the desired goal according to the optimum collision free path determined by the controller, the controller is capable of re-planning the new optimum collision free path. MATLAB simulation is

developed to verify and validate the algorithm before they are real time implemented on Team AmigoBotTM robot. The results obtained from both simulation and actual application confirmed the flexibility and robustness of the controllers designed in path planning.

Facundo Benavides, Gonzalo Tejera, Martín Pedemonte, Serrana Casella [26] proposed a model of the environment and the search algorithm is basic issues in the resolution problem. In this paper highlights the main features of Path Planning proposal for mobile robots in static environments. The path planning is based on Voronoi diagrams, where obstacles in the environment are considered as the generating points of the diagram, and a genetic algorithm is used to find a path without collisions from the robot initial to target position. This work combines some ideas presented by Roque and Doering, who use Voronoi diagrams for modeling the environment, and other ideas presented by Zhang et al. who adopt a genetic algorithm for computing paths on a regular grid based environment, considering certain quality attributes. The main results were probed both in simulated and real environments.

Buniyamin N., Wan Ngah W.A.J., Sariff N., Mohamad Z. [27] presented an overview of path planning algorithms for autonomous robots and then focused on the bug algorithm family which is a local path planning algorithm. Bug algorithms use sensors to detect the nearest obstacle as a mobile robot moves towards a target with limited information about the environment. 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.

They also introduced 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. Further this approach is then compared with other existing selected local path planning algorithm for total distance and a guarantee to reach the target.

Azali Saudi and Jumat Sulaiman [28] attempts to solve robot path planning problem iteratively using numerical technique. It is based on the use of Laplace's Equation to compute potential function in the configuration space of a mobile robot. This paper proposed a block iterative method known as Four Point-Explicit Group via Nine-Point Laplacian (4EG9L) for solving robot path planning problem. By employing a finite-difference technique, the experiment shows that it able to generate smooth path between the start and goal points. The simulation results show that 4EG9L performs faster than the previous method in generating path for mobile robot motion

Aditya Mahadevan and Nancy M. Amato [29] they investigated how agents can work cooperatively to perform tasks, plan paths in dynamic environments, or influence another group of agents to locations in an environment. Their goal was create a framework for simulating and controlling communities of characters that can dynamically interact with each other and their environment. There are many important applications of this system, ranging from civil crowd control (e.g., planning exit strategies from buildings or sporting event venues), to education and training (e.g., providing museum exhibits or training systems), to entertainment (e.g., interactive games).

Wang-bao Xu, Xue-bo Chen, Jie Zhao, Xiao-ping Liu [30] proposed a function segment artificial moment method for sensor-based path planning of a single robot in complex environments. Similar to the existing artificial moment method, an attractive point is obtained first at each step for guiding the robot to move along a shorter path. Then, the robot moves one step to the next position under the guidance of the attractive point and the control of an artificial moment motion controller. Different from the existing one, attractive function segments are used for attractive points and the artificial moment motion controller is improved. As each attractive function segment can be used for several steps, the computational burden for attractive points is reduced considerably. Furthermore, positive and negative key obstacle function segment sets are proposed to determine whether the target is reachable when a new attractive function segment is required. If the target is reachable, then, a motion direction for the robot is determined by

using the key obstacle function segment sets. Simulation results indicate that the proposed method is effective and can yield better solutions in complex environments.

Marco A. Contreras-Cruz, Victor Ayala-Ramirez, Uriel H. Hernandez- Belmonte [31] proposed an evolutionary approach to solve the mobile robot path planning problem. The proposed approach combines the artificial bee colony algorithm as a local search procedure and the evolutionary programming algorithm to refine the feasible path found by a set of local procedures. The proposed method is compared to a classical probabilistic roadmap method (PRM) with respect to their planning performances on a set of benchmark problems and it exhibits a better performance. Criteria used to measure planning effectiveness include the path length, the smoothness of planned paths, the computation time and the success rate in planning.

Oscar Montiel, Ulises Orozco-Rosas, Roberto Sepúlveda [32] proposed a method called Bacterial Potential Field (BPF) ensures a feasible, optimal and safe path **i**n environments with static and dynamic obstacles for a mobile robot (MR). This novel proposal makes use of the Artificial Potential Field (APF) method with a Bacterial Evolutionary Algorithm (BEA) to obtain an enhanced flexible path planner method taking all the advantages of using the APF method, strongly reducing its disadvantages. Comparative experiments for sequential and parallel implementations of the BPF method against the classic APF method, as well as with the Pseudo-Bacterial Potential Field (PBPF) method, and with the Genetic Potential Field (GPF) method, all of them based on evolutionary computation to optimize the APF parameters, were achieved. A simulation platform that uses an MR realistic model was designed to test the path planning algorithms. In general terms, it was demonstrated that the BPF outperforms the APF, GPF, and the PBPF methods by reducing the computational time to find the optimal path at least by a factor of 1.59.

Alejandro Hidalgo-Paniagua a,n, Miguel A. Vega-Rodríguez a,n, Joaquín Ferruz b, Nieves Pavón c [33] proposed a Multi-Objective approach based on the Shuffled Frog-Leaping Algorithm (MOSFLA) to solve the PP problem. This algorithm is inspired on the frogs' behavior in nature. In their problem definition they considered three different objectives: the path safety, the path length, and the path smoothness. The last one is a

very important objective in mobile robotics since it is directly related to the energy consumption. Furthermore, eight realistic scenarios have been used for the paths calculation and the assessment of the proposal. In order to compare the obtained results, they also used the well-known Non-dominated Sorting Genetic Algorithm II (NSGA-II), which is the most commonly used algorithm for the PP problem in a multi-objective way. With respect to the results evaluation, on the one hand, they used specific quality metrics. On the other hand, to demonstrate the statistical relevance of the obtained results they performed an in-depth statistical analysis. Finally, the study shows that the proposed MOSFLA is a good alternative to solve the PP problem.

Peter Brass, Ivo Vigan , Ning Xu [34] consider the problem of finding the shortest path for a tethered robot in a planar environment with polygonal obstacles of n total vertices. The robot is attached to an anchor point by a tether of finite length. The robot can cross the tether; i.e., the tether can be self-intersecting. Neither the robot nor the tether may enter the interior of any obstacle. The initial tether configuration is given as a poly line of k vertices. If the tether is automatically retracted and kept taut, we present an $O(kn2 \log n)$ time algorithm to find the shortest path between the source and the destination point. This improves the previous $O(lkn3)$ time algorithm, where l is the number of loops in the initial tether configuration. If the tether can only be retracted while the robot backtracks along the tether, we present an algorithm to find the shortest path in $O((n + \log k) \log n)$ time.

1.4 OBJECT AND SCOPE OF PRESENT RESEARCH WORK

Aim of the present project is the development of necessary algorithm and software based on E-Bug algorithm to navigate a mobile robot in presence of static obstacles from a starting point to a destination or goal point with the use of a range sensor. A Parallax mobile robot (Boe-Bot), existing in the Robotics laboratory of Production Engineering Department of Jadavpur University, has been used for the present project. A previous project was done with the same Boe-Bot robot using PointBug algorithm. Here, the aim of the project is to develop a program based on E-Bug algorithm which is an improved PointBug algorithm. In E-Bug algorithm some major limitations or deficiencies of PointBug algorithm are solved for obtaining near optimal path for a mobile robot. Ultrasonic range sensor, fitted with the Boe-Bot mobile robot has been used to detect obstacles.

The algorithm determines the next point to move for the robot towards target from any current point, which is the starting point at the beginning. The next point to move is dependent on the output of an ultrasonic range sensor that gives the distance of the nearest obstacle from the sensor, and thereby also detects the sudden point (sudden change in distance from the sensor to the nearest obstacle). The sudden change of range sensor output is considered that reading of distance either it is increasing or decreasing by a considerable amount, which is to be defined properly. The details of the procedure will be described in chapter 3.

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

(a) To set up an arrangement for the workspace consisting of a Boe-Bot mobile robot, Ping ultrasonic range sensor and obstacles on a suitable worktable, having a marked boundary.
(b) To mount the Ping ultrasonic range sensor on a bread board fixed on the Boe-Bot mobile robot system, and make necessary hardware connection to connect it to the Basic Stamp microcontroller through its input-output ports (pins).

(c) To develop a simplified algorithm based on E-Bug algorithm for moving the mobile robot from a start point to a target point by detecting the presence of an obstacle, if any, in the path by ultrasonic range sensor.
(d) To develop a program in PBASIC language for the Boe-Bot mobile robot for producing necessary movements of the mobile robot in presence of static obstacles for moving from a starting point to a target point using the algorithm.
(e) To run the program for different layout of workspace for testing the algorithm.

Further scope of the present project includes development of path planning algorithms using other sensors including vision system in presence of both static and dynamic obstacles to obtain an optimized path from a source to a goal point.

Chapter-2

DIFFERENT PATH PLANNING TECHNIQUES FOR MOBILE ROBOTS

2.1 CLASSIFICATION OF DIFFERENT PATH PLANNING TECHNIQUES

Generally in robotics, path planning is focused on designing algorithms that generate useful motions by processing simple or more complicated geometric models. Path planning in robotics can be divided in three main groups – motion planning, trajectory planning and planning under uncertainty. Path planning addresses the automation of mechanical systems that have sensors, actuators, and computation capabilities. In the motion planning and trajectory planning are defined as a fundamental needs in robotics that are described by algorithms that convert high-level specification of task from humans into low-level description of how to move.

Various approaches have been introduced to implement path planning for a mobile robot .The approaches are according to environment, type of sensor, robot capabilities and etc, and these approaches are gradually toward better performance in term of time, distance, cost and complexity. Mobile robot navigation problem can be divided into three subtask namely mapping and modeling the environment, path planning and path traversal with collision avoidance. Mobile robot navigation problems cannot be decomposed into fixed subtasks because the navigation problem varies according to approach used to solve the problem. As an example, the bug algorithm [15] solves the navigation problem without need to map and model the environment and only response to the output from contact sensor. Mobile robot path planning can be classified according to type of environment, algorithm and completeness. So according to environment it is classified as static and dynamic, according to algorithm it is classified as local and global and according to completeness it is classified as complete and heuristic.

2.1.1 Path planning in static environment

The static path planning refers to environment which contains no moving objects or obstacles other than a navigating robot. In a static or known environment, the robot knows the entire information about the environment before it starts travelling. Therefore the optimal path could be computed offline prior to the movement of the robot. The path planning techniques for a known and static environment are relatively mature .The

method applied a reordering operator for performance enhancement and the algorithm was capable of determining a near-optimal solution.

2.1.2 Path planning in dynamic environment

Dynamic path planning [35] refers to environment which contains dynamic moving and changing object such as moving obstacle. In case of path planning in dynamic environment, the status and the movement of the obstacles change continuously in the map. Moving obstacles in the dynamic environment increases the difficulty of planning path for the robots in the map. The robot has to use sensors acquiring the information of surrounding environment and do on-line real-time path planning. The planning time for the robot should be short because the robot needs a sufficient time intervals to adjust its movement in order to avoid the coming obstacles.

Complexity and uncertainty increases with the number of the dynamic obstacles. Therefore, traditional path planning algorithms do not perform well in dynamic environments. Robot path planning in dynamic environment is thereby an issue for further research.

2.1.3 **Global path planning**

Global path planning [36] is a path planning that requires robot to move with prior 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. 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.

2.1.4 Local path planning

Local path planning [37] 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.

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 sense 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 [3] is the one of the well known local path planning technique. In this path planning method, the robot is considered as a particle moving under influence of an artificial potential produced by the goal configuration and the obstacles.

2.2 FAMILY OF BUG ALGORITHMS

One of the earliest algorithms is Bug algorithm, which plans direct path from source to destination until it faces an obstacle. Bug algorithms are well known mobile robot navigation method for local path planning with minimum sensor and simple algorithm. The variations of bug algorithms showed the effort toward shorter path planning, shorter timing, simpler algorithm and better performance.

2.2.1 BUG-1 algorithm

Bug-1 is the earliest obstacle avoidance algorithm [15], it is easy to tune and does not suffer by local minima however it takes the robot far away from the goal in some scenarios. In this algorithm, the robot after detecting an obstacle starts following the edge of obstacle until it reaches to the point from where the robot started following the edge. It simultaneously calculates the distance from current position to destination and finally stores the point having minimum distance. This point, after one complete cycle of the robot is considered as leaving point. The robot restarts following the edge until it reaches to the calculated leaving point. After avoiding obstacle, robot computes new path from the leaving point to destination using straight line equation. The robot follows that straight line until it reaches the destination or another obstacle encountered. One of the common drawbacks of Bug-1 algorithm is, when the robot is following the edge of obstacle 1, it may collide with a neighboring obstacle 2 in case when the later is in very close proximity to the first obstacle or the gap between them is less than the width of the robot.

2.2.2 BUG-2 algorithm

Bug-2 algorithm [5] is an improved version, which generates initial path from source to destination and stores slope of this path in its move to goal behavior. The behavior of the robot is changed to obstacle avoidance when an obstacle is encountered, where the robot starts following edge of the obstacle and continuously calculates slope of the line from its current position to the destination. When this slope becomes equal to slope of initial path (from source to destination), the behavior of the robot is changed to move to goal. Therefore, the robot follows single non-repeated path throughout its

trajectory. Bug-2 algorithm is more efficient than Bug-1 algorithm as it allows the robot to reach the destination in less time following a short trajectory. Both Bug1 and Bug2 algorithms demand minimum memory requirements. However, they do not have capability to make optimum use of sensors data for generation of short paths.

2.2.3 ALG-1 algorithm

Alg-1 improved Bug-2 weakness that is it can trace the same path twice by storing the sequence of hit points occurring within an actual path to the goal. These storing data are used to generate shorter paths by choosing opposite direction to follow an obstacle boundary when a hit point is encountered for the second time.

2.2.4 ALG-2 algorithm

The Alg-2 algorithm is the improved form of Alg-1 algorithm both of these are introduced by the same researcher. It ignores the m-line of Bug-2 algorithm with new leaving condition. The Alg1 and Alg2 [6] still face a reverse procedure problem where after encountering a visited point that causes a loop in a mobile robot follows an uncertain obstacle by an opposite direction until it can leave the obstacle.

2.2.5 REV-1 & REV-2 algorithm

The problem of reverse procedure in Alg-1 and Alg-2 was solved by Horiuchi by introducing a mixing reverse procedure with alternating following method to create shorter average bound of path length and named the algorithm as Rev1 and Rev2 [9]. Alternating following method is defined as independently, if a robot always changes a direction following an uncertain obstacle alternatively, the robot arrives at a destination earlier on average and there will decrease probability for the robot to join a loop around a destination.

2.2.6 DIST-BUG algorithm

This algorithm is based on distance, in which robot moves from source to destination on path having minimum distance. When robot faces an obstacle in path, it starts following the edge of obstacle simultaneously; it calculates the distance of

destination from each point. The point with the minimum distance is known as leaving point. When it finds the leaving point during its motion around an obstacle, it generates a new path and starts following it until reaches to the destination. The DistBug algorithm [7] is a local path planning algorithm that guarantees convergence and will find a path if one exists. It requires its own position by using odometry, goal position and range sensor data. To guarantee convergence to the target, the DistBug algorithm needs a small amount of global information for updating d-min line and for determining that the robot completed a loop around an obstacle. The value of d-min can be extracted directly from the visual information. This guarantee convergence using updating d-min value makes problem in determining accuracy because the value of d-min is taken from direct global visual information.

2.2.7 TANGENTBUG algorithm

The TangentBug [9] is another variation of DistBug that improves the Bug2 algorithm in that it determines a shorter path to the goal using a range sensor with a 360 degree infinite orientation resolution .Tangent Bug incorporates range sensors from zero to infinity to detect obstacles. When an obstacle is detected, the robot will start moving around the obstacle and will continue its motion toward target point routine as soon as it has cleared the obstacle. During following boundary, it records the minimal distance to target about d-min line which determines obstacle leaving and reaching condition .The robot constructs a local tangent graph (LTG) based on its sensors' immediate readings. The LTG is constantly updated and it is used by the robot to decide the next motion. The disadvantage of this algorithm is requiring robot to scan 360 degree before making decision to move to the next target.

2.2.8 POINTBUG algorithm

Another efficient algorithm is PointBug algorithm [27]. It navigates a point of robot in planar of unknown environment which is filled with stationary obstacles of any shape. It determines where the next point to move toward target from a starting point. The next point is determined by output of range sensor which detects the sudden change in distance from sensor to the nearest obstacle .Its principle strategy is few different than

others algorithms when it introduces the new notion "*sudden points*". The *sudden point* is a point where a sudden change in distance of sensor's range is detected. There are three possible changes on the detected distance: (a) from infinite to some distance in some vertex of obstacles, (b) from some distance to infinite distance in other vertex of obstacles, (c) from some distance to some distance when an obstacle hides a part of itself or of another obstacle. The robot is capable to scan the environment using range sensor by rotating itself from 0 degree up to 360 degree at a constant speed. The first direction of robot is facing the target point where it starts searching for the first sudden point by rotating to right or left. After, the robot repeats these two actions: (1) Moves towards next sudden point; (2) rotates in the direction of *dmin-line*1 for searching the next sudden point. The dmin line is the shortest distance in one straight line between current sudden point and target point and its value is always recorded every time the robot reaches new sudden point The robot always ignores the sensor reading at rotation of 180 degree to avoid detection of previous sudden point making the robot return to previous sudden point from its current point. If there is no sudden point found within a single 360 degree rotation, the target is considered unavailable and the robot stops immediately. When the robot searches sudden points it ignores sensor reading at rotation of 180° to avoid detecting previous sudden points.

Dynamic Point Bug algorithm [35] uses another strategy; the robot advances and adjusts its direction continuously to the target. When the robot encounters an obstacle it moves right or left to avoid it, then it continues its path toward the target. In addition of the amount of calculation made the continuous adjustment of direction can lead to infinite oscillation in the case of local minimums or spiral.

Limitations of PointBug algorithm:

PointBug algorithm has several limitations [38]. Out of these the most important limitations are focused here these are:

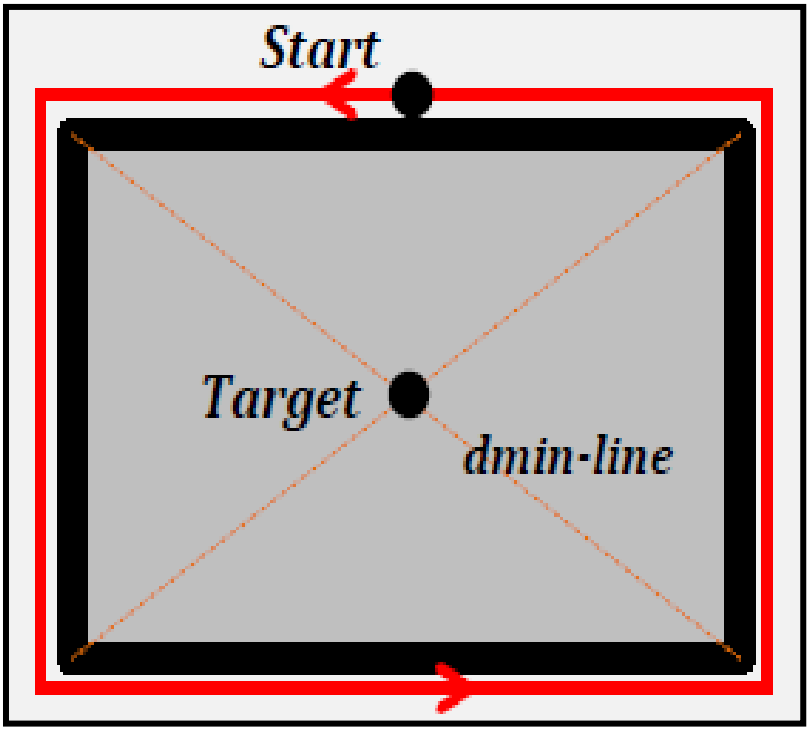

Figure 2.1 A case of an infinite loop produced by PointBug algorithm

a) Problem of infinite loop for repeated visit of some sudden points:

There are no tests in PointBug algorithm if a sudden point was already treated as it does not record visited sudden points. This situation can produce infinite loops in case where the target is surrounded by uniform boundary obstacle, see Figure 2.1.

b) Minimizing the angular deviation relative to target direction does not always produce the optimal path:-

Many algorithms including PointBug choose the next points to minimize the deviation angle from target direction. This may not produce the optimal path, as illustrated bellow in fig. 2.2

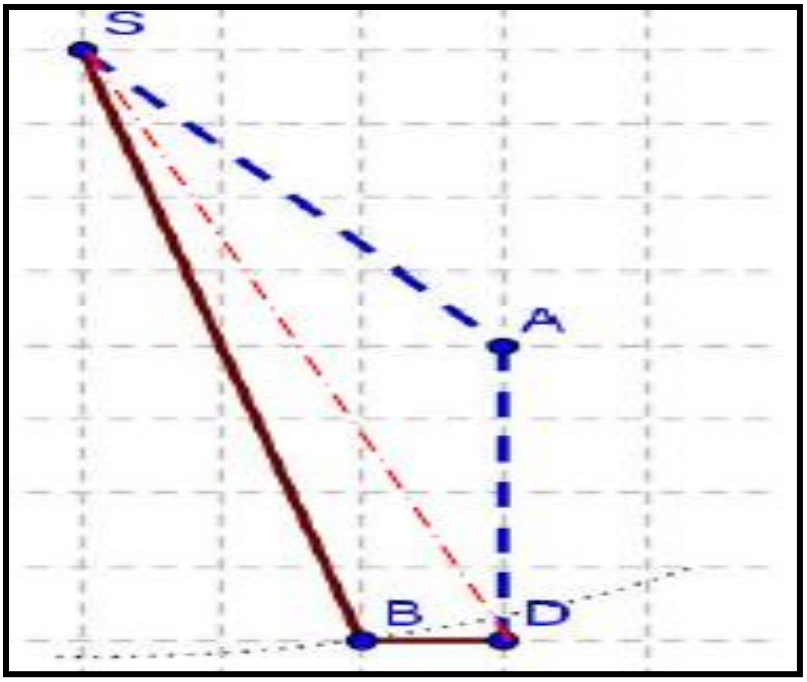

Figure 2.2 Minimizing the angular deviation does not produce the shortest path

In this Figure 2.2 [39], the deviation angle formed by the point (A) [SD, SA] is 24.4° and the path length (S)-(A)-(D) = 9 cm, and the deviation angle formed by (B) [SD, SB] is 14.6° and the path length (S)-(B)-(D) = 9.246 cm. So the near optimal path is 9 cm for greater the deviation of angle. But according to PointBug (or other algorithm with the same principle) the chosen path length is 9.246 cm which is not near optimal although here the deviation angle is less. So, the greater deviation angle formed by the point (A) [SD, SA] gives the shorter path.

Chapter-3

IMPROVED PATH PLANNING ALGORITHMS TO OVERCOME THE LIMITATIONS OF POINTBUG ALGORITHM

3.1 K-BUG ALGORITHM

One of the most popular improved PointBug algorithms is K-Bug algorithm [40]. It consists of two forms. The principle of the first form of K-Bug algorithm is similar to PointBug but instead of using angle of rotation, it just selects the nearest vertex until reaching the target. The second form of K-Bug is an algorithm, where it requires prior full knowledge of the environment. It considers the line (source, target), that crosses the obstacles and cut its edges in a pair points; "*entry* and *exit*". The side with smallest perimeter is selected, and then the farthest vertex from the line (current vertex, target) will be inserted to the path. These actions are repeated with each new vertex until avoiding all collisions and arriving to the target.

K-Bug algorithm limitations:

a. Problem of choosing the good sensor range:

In its first form K-Bug algorithm chooses the nearest vertex of each encountered obstacle, but if the range of used sensor is limited, the robot may not detect some vertex. In the worst case, it cannot detect any vertex. As result, this algorithm will be unusable in a real world at least in its present form, as shown in figure 3.1.

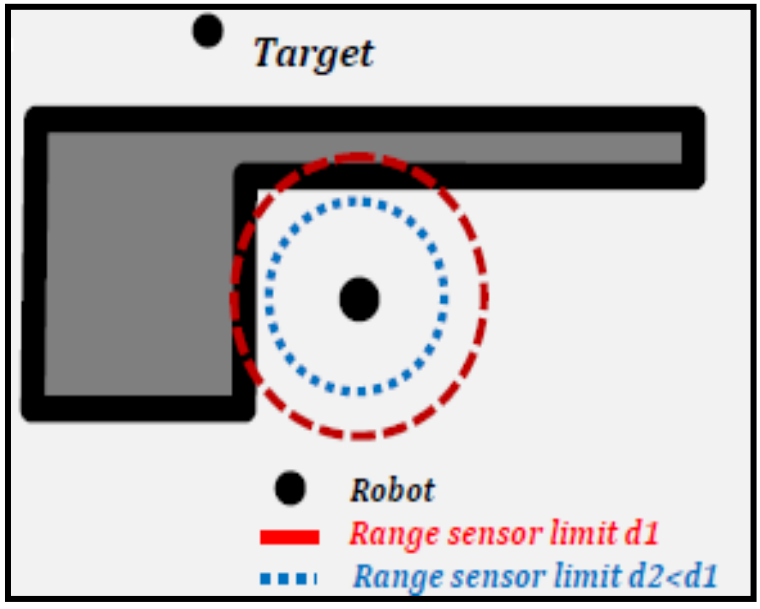

Figure 3.1: K-Bug is unable to choose the advancing direction

b. Problem of choosing the side with smallest perimeter to avoid obstacle:

This strategy used by K-Bug gives good results in many cases, but they depend of environment. So using this strategy in an unknown environment is like choosing the direction randomly. Figure 3.2 shows an example [39] where K-Bug chooses the worst path in spite of the full environment knowledge.

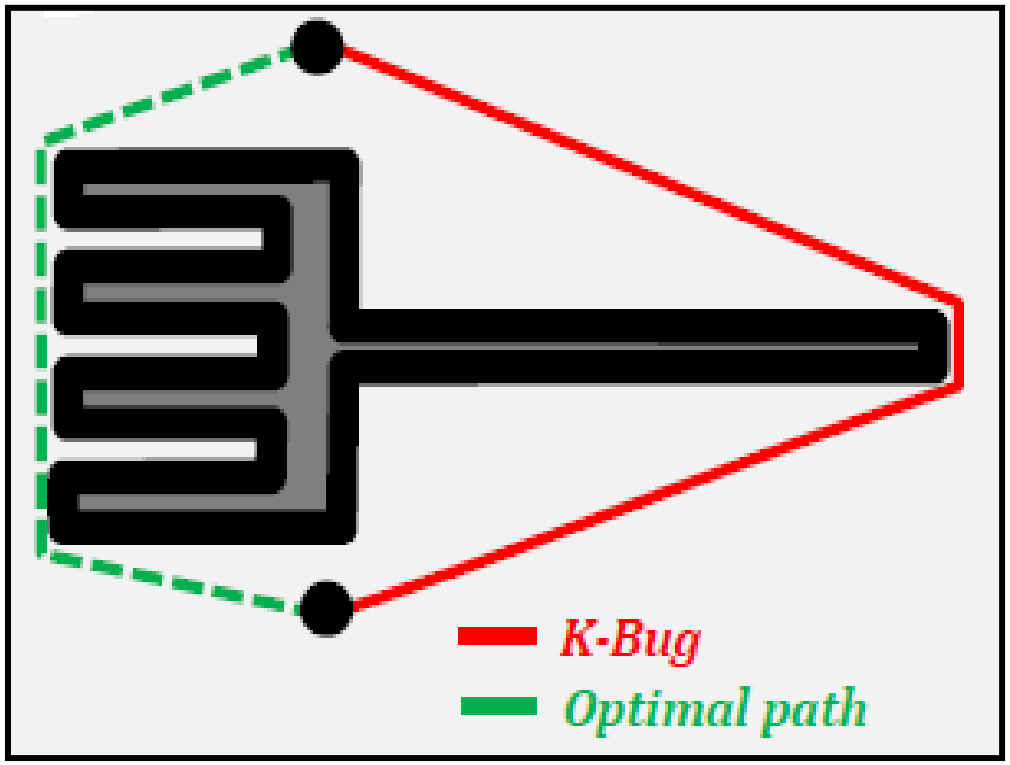

Figure 3.2 Path generated by k-bug is not optimum

3.2 P* ALGORITHM

P*(P STAR) [38] is a new sensor based path planning algorithm for autonomous robot in an unknown environment. The principle strategy of this algorithm is based on PointBug algorithm. P* algorithm is improved and modified form of PointBug algorithm to overcome some deficiencies or limitations faced by PointBug algorithm.

Some problems of Point Bug and K-Bug were presented and solved by P* algorithm. This algorithm uses the similar strategy of Point Bug but it eliminates the limit of 180° when searching for new sudden point. This limit can lead to an infinite loop. P* practices a principle like Brute Force Searching algorithm (BFS) where it saves the visited sudden point list, so the examination of all sudden points is guaranteed. Another improvement made by P* is the boundary following in same cases to minimize the number of sudden points.

For the solution of endless cycles and double treatment of sudden points, the condition of 180° bound was removed. The points above 180° will not be ignored. Also, the sudden points were recorded and just ignore the sudden points already treated and their sons as well (all their succeeding sudden points). The algorithm follows the same strategy with all points. So, the path generated will be the same as the previous one. As a result the current path will be ignored completely. When the robot finds a treated sudden point once again, it ignores this point and continues the treatment of the succeeding sudden point. If there is no sudden point, the robot return to previous one, until reaching the target or no more sudden point can be found from the first one.

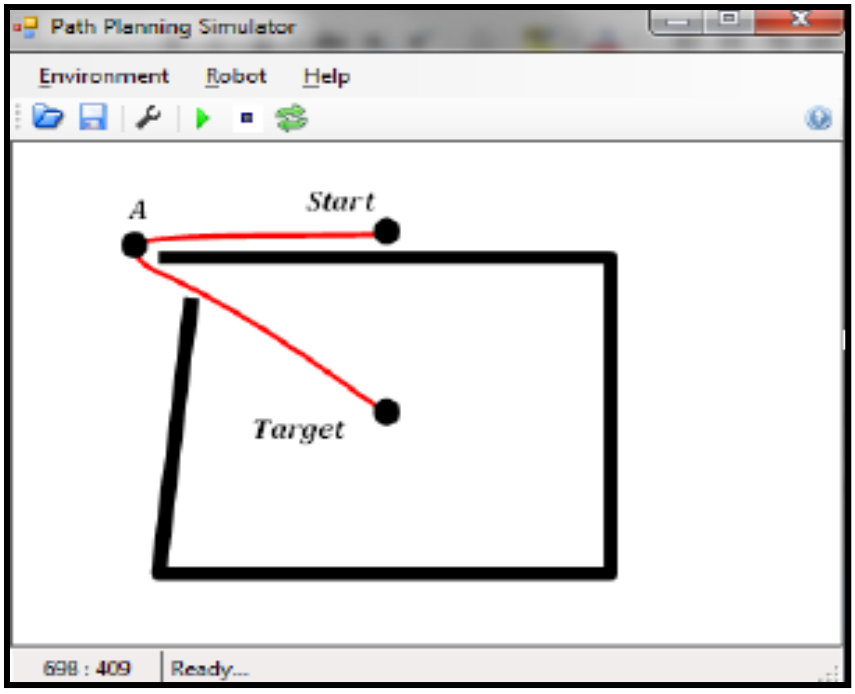

Figure 3.3 P* algorithm gives a solution of endless cycle

In figure 3.3 for P* algorithm [38], the mobile robot finds two sudden points and moves towards the sudden point which makes least distance from current position to goal position. In this way the problem of endless cycle in PointBug algorithm is solved by P* algorithm.

3.3 E-BUG ALGORITHM

E-Bug algorithm [39] is also an improved form of PointBug algorithm, where some modification is made to overcome some of the limitations faced by PointBug algorithm. In E-Bug the definition and selection criteria of sudden points are changed.

A sudden point is a point where the robot detects a sudden difference on the distance toward the obstacle .This definition contains some failures, for example: In the real world robot there is no infinite range sensors. So using the original definition, we can only detect one type of sudden point. When the robot is very close to an obstacle, it will detect a very high number of sudden points, but in fact there is no sudden point. The modified definition is as follows: a sudden point is a point where the difference on distance exceeds a certain defined value just in the limit of detection range.

In E-Bug algorithm the objective is to minimize the sum of sub-paths leading to the target. But by the definition in an unknown environment only a limited part of environment is known. The robot knows only a subset of sudden points. So the local sub path lengths can only be minimized for the success of the algorithm. This last is equal to the sum of the old distances plus the distance from the current point to the next sudden point and from the next sudden point to the target. Consequently, the next sudden point is chosen and aimed at minimizing this sum (of these two next distances). At each time, E-Bug handles the visible sudden points only which made it more adequate for the real-time robots. The computations are made only when reaching the next sudden point. So comparing E-Bug algorithm with TangentBug or Dynamic PointBug algorithms E-Bug algorithm gives more satisfactory or near optimal path with more intelligence.

Description of E-Bug algorithm formalization:

As defined, sudden points can exist only in pinnacle of obstacles. Consequently, it can’t exist any path between two consecutive sudden points in the boundary of any obstacle. So the optimum path must relay a set of existing sudden points (if they exist). As a result, the main problem of searching the optimal path will be reduced to find the optimal path passing by a set of the existing sudden points. At any point the robot can detect only a subset of sudden points as shown in Figure 3.4.

The main goal of the path-planning is to minimize the global path length**,** but to get the global path length minimal all sub-paths Pi must be known, which contradict with problem definition. So, the path length is minimized in the detected area using the current set of sudden point. It was assumed that it doesn't exist any obstacle between the next point and the target (all point have the same probability). The algorithm calculates the sum of two distances: from the current position of robot to the next sudden point, and from this next sudden point to the target. The aim of the algorithm is to minimize the previous sum each time until reaching the target.

Now, the optimal path never contains a cycle. In other expression a point can't exist twice in the optimal path.

A set of points (*S*), (*C*) and (*R*) are considered here:
-(*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.

This all set of points formulate a closed path will be eliminated. The robot will simply ignore any point previously rejected.

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.

This action is applied for all next sudden point except the rejected ones. 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.

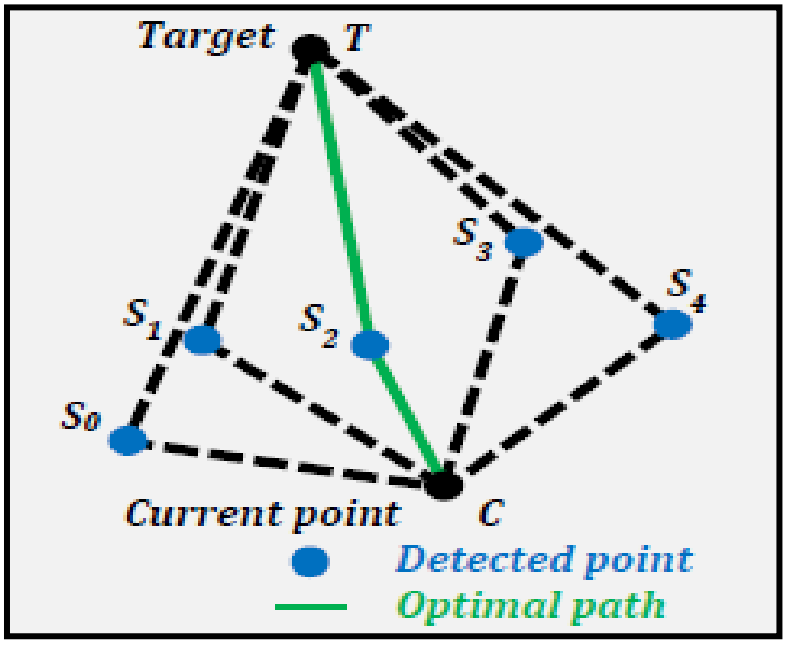

Figure 3.4 Strategy of E-Bug algorithm

The example above (Figure 3.4) shows clearly the strategy of E-Bug algorithm .The robot will choose the point (S2) since the sum of distances [C, S2] and [S2, T] is the minimal. Arriving at any sudden point, the proposed algorithm tests all accessible sudden points (S0 to S4) has shown in Fig. 3.4 [39]. Between all these points, E-Bug selects the point S2 because it provides the shortest path to the target.

([C,S2]+[S2,T]<[C,S1]+[S1,T]<…<[C,S4]+[S4,T]), S2 will be added to the current shortest path (C). When S2 is already selected, it will be added to the rejected set (R) and ignored afterward. When no further sudden point can be found the robot turns back and continues with the next one after adding the current point to the rejected set (R).

Therefore, the test of all accessible sudden points is guaranteed, and if the path exists it must be found consequently the convergence of our algorithm is assured.

Arriving at any sudden point, the robot chooses the shortest sub path in the visible part of the environment, and assumes it doesn't exist an obstacle in the invisible part of the environment (between the last sudden point and the target), until reaching this part.

Thus the proposed algorithm guarantees that the current sub path is optimal (precedent, current, and the next sudden point).

The algorithm as shown below:

E-Bug algorithm:

Begin

1. From the current position enumerate all sudden points can be detected.
   If can reach the goal End with success.
   If no new sudden points is detected ($C=\phi$) 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 ($Si€C$) expect the rejected ones ($Si€R$) 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 provides the minimal distance.
   End if
   Go to 1.

End.

In the present project, a simplified form of E-Bug algorithm has been used for a Parallax Boe-Bot mobile robot kit fitted with an ultrasonic range sensor to find a near optimal and collision free path from a start point to a goal point. The details of the procedure will be described in section 4.4 of chapter 4.

Chapter-4

IMPERIMENTATION OF E-BUG ALGORITHOM WITH A PARALLAX BOE-BOT MOBILE ROBOT KIT FITTED WITH AN ULTRASONIC RANGE SENSOR

4.1 BOE-BOT SYSTEM HARDWARE & SOFTWARE

Parallax Boe-Bot mobile robot kit:

Parallax Boe–Bot (short for Board of Education Robot) consists of a main circuit board (the Board of Education), a plug–in microcontroller, two servo motors to drive the wheels, a bread board and an aluminum chassis that the parts bolt onto.

The green detachable main circuit, mounted on the top of the robot is called the Board of Education. The microcontroller which plugs into a socket on the green circuit board is called the BASIC Stamp .The BASIC Stamp is programmed in PBASIC. The rear wheel is a drilled polyethylene ball held in place with a cotter pin. Wheels are machined to fit on the servo spline and held in place with a screw. These servomotors are connected with the P14 and P15 pins on the Board of Education carrier board. To load the program onto microcontroller, bread board is connected to debug terminal using serial interface. The BASIC Stamp is easy to program. The parallax Boe–Bot is small, approximately four inches wide, and runs on four AA batteries. At the front of the chassis, there is a gripper installed for gripping the objects. Separate servomotor is used to operate the gripper. Pin P13 is connected with this servomotor [41].

The PING ultrasonic range sensor [41] can be attached on the Boe-Bot. Fig 4.1 shows a photo graphic view of Boe-Bot robot with gripper and fitted with ultrasonic range sensors.

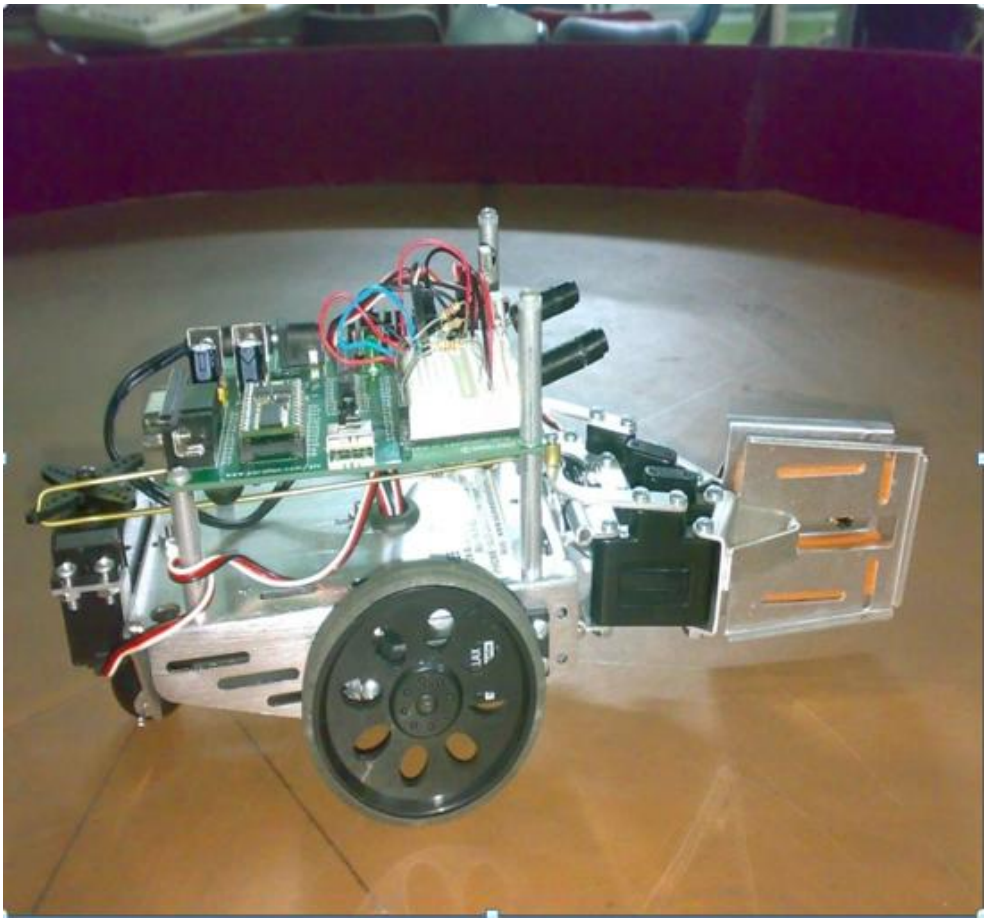

Fig. 4.1 Photographic view of Boe-Bot with gripper

4.1.1 Board Of Education Carrier Board

The Board of Education carrier board [41] along with the BASIC stamp 2(microcontroller) and other electronic components is shown in fig. 4.2. It is the main board that contains all the electronic components including the microcontroller, bread board and necessary parts for connecting servomotors and various sensors. Board of Education carrier board power requirement is 6 to 9 VDC and operating temperature is +32 to +185 °F (0 to +70 °C).

The BASIC Stamp module plugs into the Board of Education carrier board. The Board of Education makes it easy to connect a power supply and serial cable to the BASIC Stamp module. It also makes it easy to build circuits and connect them to the BASIC Stamp.

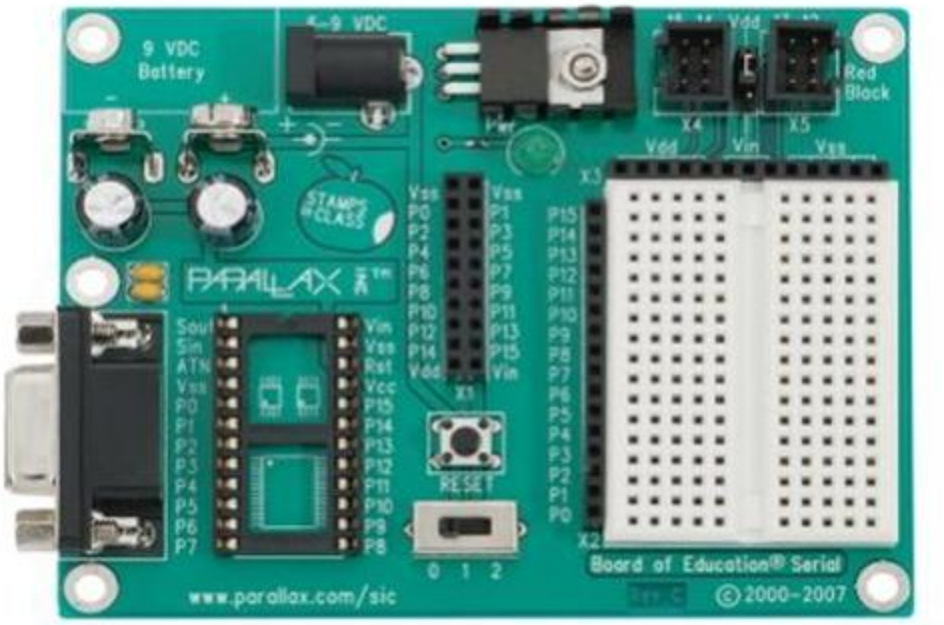

Figure 4.2 board of education carrier board

4.1.2 Basic Stamp 2 Microcontroller Module

The BASIC Stamp 2 microcontroller module serves as the brains inside of electronics projects and applications that require a programmable microcontroller. It is able to control and monitor timers, keypads, motors, sensors, switches, relays, lights, and more. Programming is performed in an easy-to-learn language called PBASIC. Its small form factor requires very little space and non-volatile memory holds up to 500 instructions even without power. This is shown in fig.4.3.

The System software that has been used for development and execution of the necessary application program in PBASIC to run on the microcontroller of the Boe-Bot system is BASIC Stamp Editor (version 2.0). This Editor program has been run on a separate computer system (PC or laptop), which is connected to the microcontroller through serial interface.

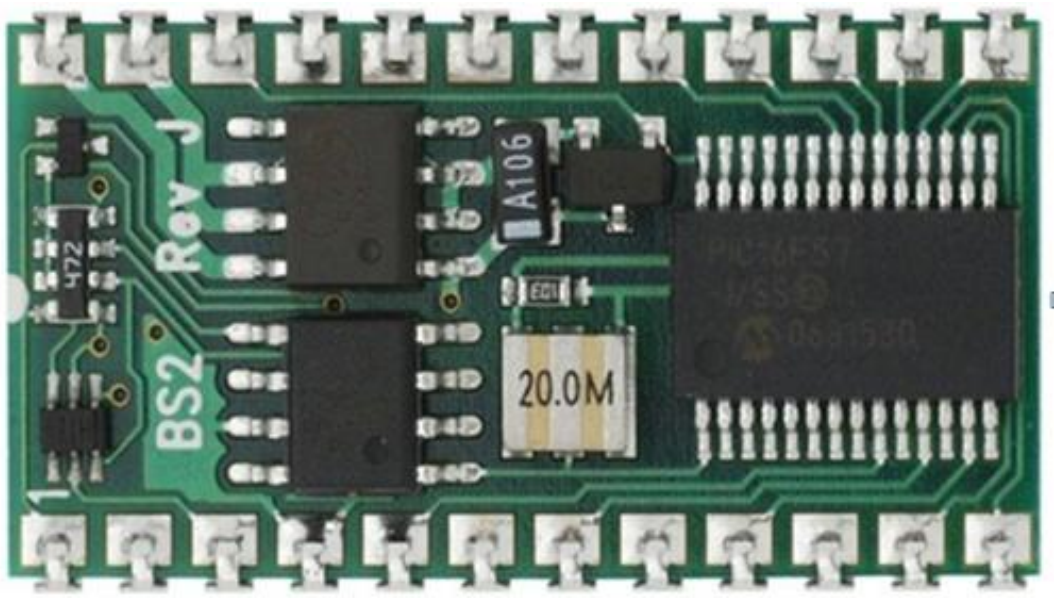

Fig.4.3 Microcontroller (Basic Stamp 2)

4.1.3 Parallax Servo Motors

The Parallax continuous rotation servo motor [41] shown in fig 4.4 is ideal for robotic products that need a geared wheel drive or a 360 degree rotation geared motor. The Parallax continuous rotation servo output gear shaft is a standard Futaba configuration. Continuous rotation servos receive the same electronic signals, but instead of holding certain positions, they turn at certain speeds and directions. BASIC stamp send the same message over and over again to control the servo motor speed and direction. The servo motors for the wheels are connected to pin numbers 14 and 15 (P14 and P15) and a separate servo motor used to actuate the gripper is connected to pin number 13 (P13). Power requirement for parallax servo motors is 4 to 6VDC

.

Figure 4.4 Parallax servo motor

4.1.4 BASIC Stamp Windows Editor (version 2.0)

It is the robot control software package. This software allows us to write programs in PBASIC on computer and download them into the Boe–Bot's BASIC stamp brain (microcontroller). BASIC Stamp Editor is free software; it can be downloaded from the internet or from the Parallax CD. A program entered in the BASIC stamp Editor Window, some lines of the program is made automatically by clicking buttons on the toolbar. Other lines are made by typing them in the keyboard. After saving the program with a file name, if it is run, a download program window will appear briefly as the program is transmitted from the PC to BASIC stamp. The Debug terminal will appear on the computer monitor when the download is completed. Debug terminal shows the message which is sent by the BASIC stamp to PC. Every time by pressing and releasing the Reset button on Board of education, the program will rerun and every time a message will be displayed in the Debug terminal.

4.1.5 PBASIC Version 2.5

It is the programming language which is used to accomplish different tasks performed by the BASIC stamp and Boe-Bot.

PBASIC stands for –

- Parallax– Company that invented and makes BASIC stamp microcontrollers.
- Beginners– Made for beginners to use learn how to program computer.
- All purpose– Powerful and useful for solving many different kinds of problems.
- Symbolic– Using symbols (terms that resemble English word /phrases)
- Instruction– To instruct a computer how to solve problem.
- Code – In terms that one and one's computer understand.

Types of Variables and Defining Variables

One can declare four different sizes of variable in PBASIC as follows:

Syntax: v**ariable-name VAR Size**

There are four different sizes of variables in PBASIC:

Size	**Decimal values to be stred**
Bit	0 to 1
Nib	0 to 15
Byte	0 to 255
Word	0 to 65535 or -32768 to +32767

Main commands and functions used in the project

DEBUG

Syntax: **DEBUG Output Data**

- Function: Display information on the PC screen within the BASIC Stamp editor program. This command can be used to display text or numbers in various formats on the PC screen in order to follow program flow (called debugging).

DEBUGIN

Syntax: **DEBUGIN Input Data**

- Function: Accept information from the user via the Debug Terminal within the BASIC Stamp Editor program.

PULSOUT

Syntax: **PULSOUT Pin, Duration**

- Function: Generates a pulse on Pin with a width of duration.
- Pin is a variable/ constant / expression (0-15) that specifies which I/O pin to set low. The pin will be placed into output mode.
- Duration is a variable/ constant/expression (0-65535) that specifies the duration of the pulse. The unit of time for duration is two microsecond.

This command has been used to rotate the servomotors of wheels and gripper in the required direction for necessary time.

PULSIN

Syntax**: PULSIN Pin, State, Variable**

Function: Measure the width of a pulse on Pin described by State and stores the result in Variable.

PAUSE

Syntax: **PAUSE Duration**

Function: Pause the program (do nothing) for the specified duration

.Duration is a variable/ constant/expression that specifies the duration of the pause. The unit of time for duration is one millisecond.

DO-----LOOP

Syntax: **DO (WHILE/ UNTIL conditions) Statement(s)**

LOOP (WHILE /UNTIL conditions)

- Function: Create a repeating loop that executes the program lines between DO and Loop, optionally testing before or after the loop statements
- Condition is an optional variable / constant / expression (0 – 65535) which determines whether the loop will run or terminate.
- Statement is any valid PBASIC statement.

FOR----- NEXT

Syntax: **FOR Counter = Start value To End value {STEP step value} ---- NEXT**

Function: Create a repeating loop that executes the program lines between FOR and NEXT, incrementing or decrementing counter according to step value until the value of the counter variable passes the End value. Counter is a variable (usually a byte or a word) used as a counter Start value is a variable/ constant/expression (0-65535) that specifies the initial value of the variable (counter).

IF…THEN

Syntax: IF (condition) THEN {ELSEIF condition)}… {ELSE}…ENDIF

Function: In order to make decisions if there is different combination conditions.

GOTO

Syntax: **GOTO Address**

Function: Go to the point in the program specified by Address. Address is a label that specifies where to go.

GOSUB

Syntax: **GOSUB Address**

Function: Store the address of the next instruction after GOSUB, then go to the point in the program specified by address, with the instruction of returning to the stored address. Address is a label that specifies where to go.

SIN, COS and ATN FUNCTIONS

The Sine operator (SIN) returns the two's complement, 16-bit Sine of an angle specified as an 8-bit (0 to 255) value. The BASIC Stamp SIN operator breaks the circle into 0 to 255 units instead of 0 to 359 degrees. This unit is a binary radian or brad. Each brad is equivalent to 1.406 degrees. And instead of a unit circle, which results in fractional Sine values between 0 and 1, PBASIC Stamp SIN is based on a 127-unit circle. So, at the origin, SIN is 0. At 45 degrees (32 brads), Sine is 90. At 90 degrees (64 brads), Sine is 127. At 180 degrees (128 brads), Sine is 0 again. At 270 degrees (192 brads), Sine is - 127. This is illustrated in fig 4.7.

The Cosine operator (COS) returns the two's complement, 16-bit Cosine of an angle specified as an 8-bit (0 to 255) value, similar as sine operator.

The Arctangent operator (ATN) returns the angle to the vector specified by X and Y coordinates values. In the PBASIC Stamp, the angle is returned in binary radians (0 to 255) instead of degrees (0 to 359). Coordinate input values are limited to -127 to 127. This is also illustrated in fig 4.5.

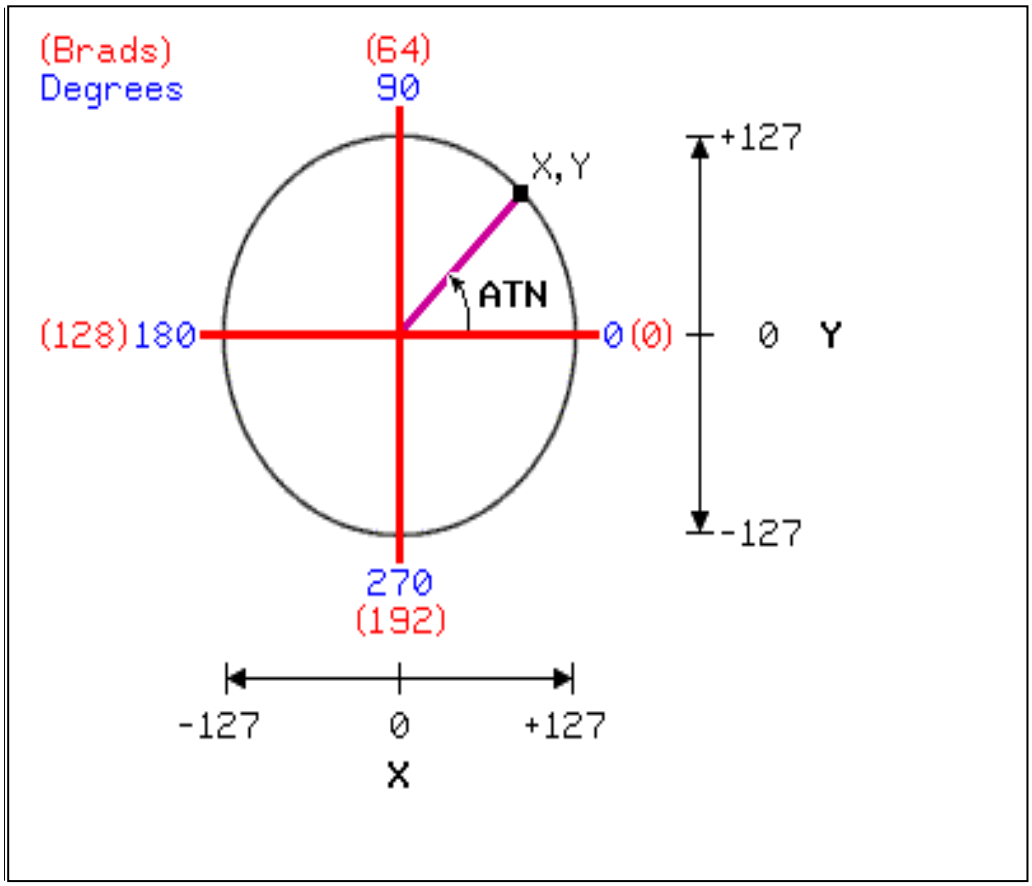

Figure 4.5 SIN, COS and ATN functions in PBASIC

4.2 BOE-BOT NAVIGATION SYSTEM

Servos are centered by running a program which sends a signal instructing them stay still. The instruction consists of a series of 1.5 ms pulses with 20 ms pauses between each pulse. As the servos are not pre-adjusted at the factory, they will instead start turning. A screw driver is used to adjust them so they stay still. In BOE-BOT, after the servo has been properly adjusted, center signal instruct it to stay still. This process is called centering the servos. Once the centering of the servo is done, a pulse width of 1.5ms causes the servos to stand still. This is done using a PULSOUT command with duration of 750 as shown in fig 4.6. It's best to only center one servo at a time, because that way we can hear when the motor stops as we are adjusting it.

The Parallax continuous rotation servo rotates full speed clockwise, when 1.3 ms pulses are sent to it, as shown in fig 4.7. Full speed ranges from 50-60 rpm. Now, a 1.3 ms pulse requires a PULSOUT command Duration argument of 650. All pulse widths less than 1.5 ms, and therefore PULSOUT Duration arguments less than 750, will cause the servo to rotate clockwise. The servo rotates full speed counter clockwise, when 1.7 ms pulses are sent to it, as shown in fig 4.8. Full speed ranges from 50-60 rpm. Now, a 1.3 ms pulse requires a PULSOUT command Duration argument of 850. All pulse widths greater than 1.5 ms, and therefore PULSOUT Duration arguments less than 750, will cause the servo to rotate clockwise.

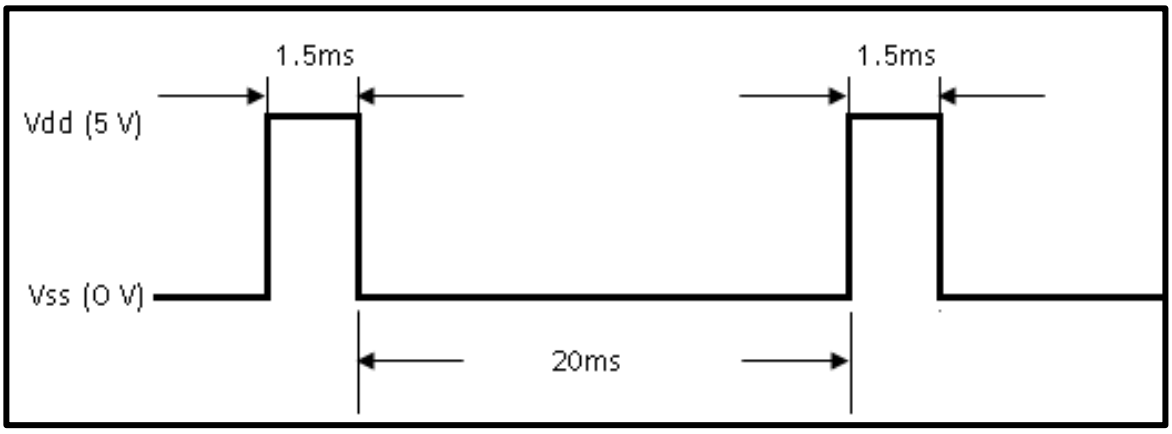

Figure 4.6 Pulse train for making the servo motor stand still

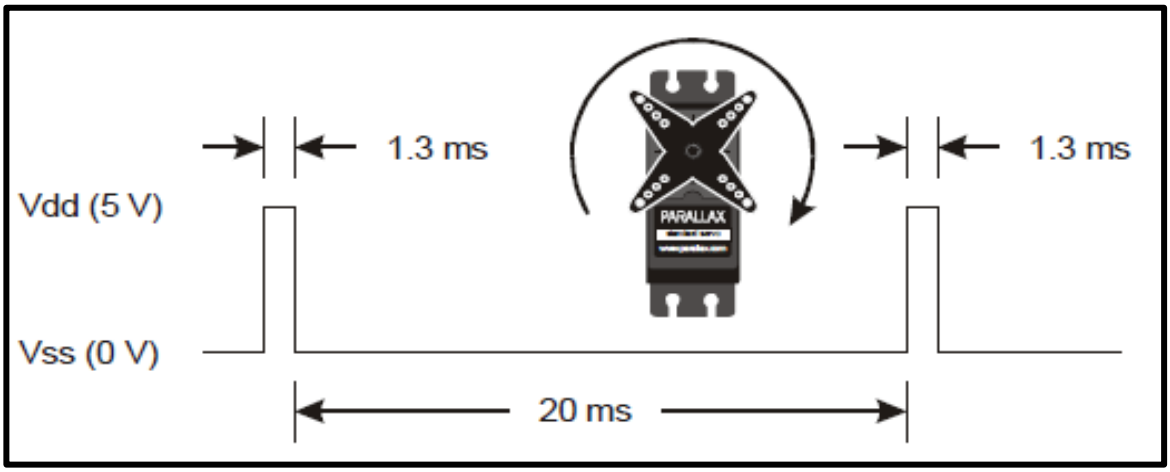

Figure 4.7 Pulse train for turning the servo motor full speed clockwise

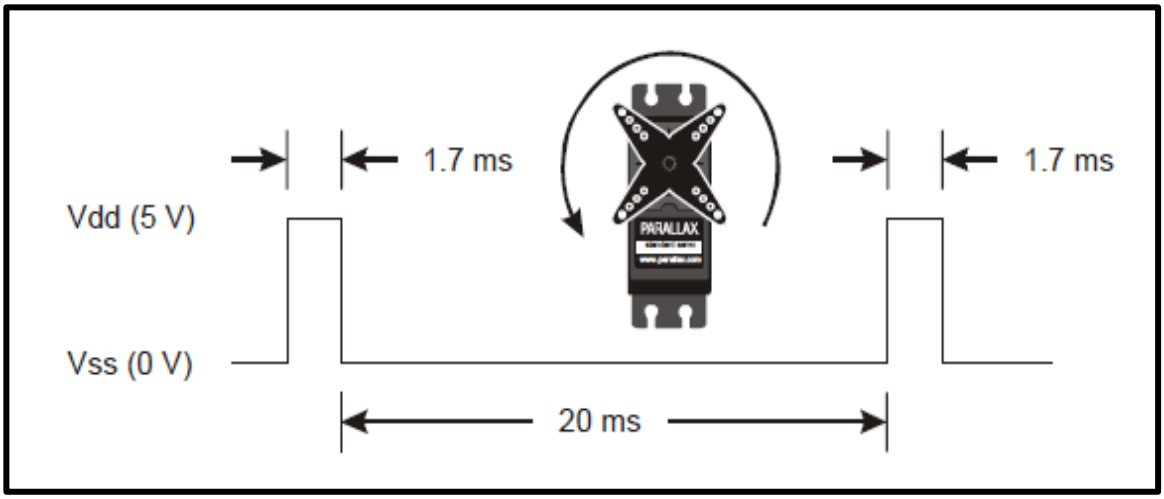

Figure 4.8 Pulse train for turning the servo motor full speed counterclockwise

Forward, backward, rotate left, rotate right and pivoting turns are the basic maneuvers of Boe-Bot which depend upon the direction of rotation of two servomotors.

Moving forward and backward:

To move the Boe-Bot forward, it‘s left wheel (driven by a servo motor connected to pin 15) will have to turn counter clockwise and right wheel (driven by a servo motor connected to pin 14) will have to turn clockwise. So following command is used to perform the forward motion at the maximum speed.

```
FOR counter = 1 TO n
PULSOUT 15, 850
PULSOUT 14, 650
PAUSE 20
NEXT
```

To move the Boe-Bot backward, it‘s left wheel will have to turn clockwise and right wheel will have to turn counter clockwise. So following command is used to perform this motion at maximum speed.

```
FOR counter = 1 TO n
PULSOUT 15, 650
PULSOUT 14, 850
PAUSE 20
NEXT
```

In both cases the value of the variable n (i.e., the number of execution of the loop) will determine the time of running the motors and hence the distance traveled by the robot. This has been explained in the system hardware section of the servo motor.

Adjusting distance and speed:

Distance covered by the Boe-Bot depends upon, how much time, the servo will run. Start value and End value of FOR---- NEXT loop command has been used to control the number of pulses that has been delivered. The End value argument also controls the time the servo will run as each pulse takes the same amount of time. PULSOUT duration argument is changed to control the speed of the servo motors. Arguments of 650 and 850 rotate the servos at maximum speed. To slow down the speed of Boe-Bot, each of the PULSOUT Duration argument has been closer to stay-still value of 750.

The distances traversed for different number of pulses have been measured at maximum speed, and are shown in table 4.1.

TABLE 4.1: Distance Traversed Against Number of Pulses Sent

OBSERVATION NO.	DURATUON ARGUMENT		NO. OF PULSES	DISTANCE (cm)
	P15 (left)	P14 right)		
1	850	650	10	5.3
2	850	650	20	8.8
3	850	650	30	12.5
4	850	650	40	16.6
5	850	650	50	20.4
6	850	650	60	24.1
7	850	650	70	28.1
8	850	650	80	32.2
9	850	650	90	36.5

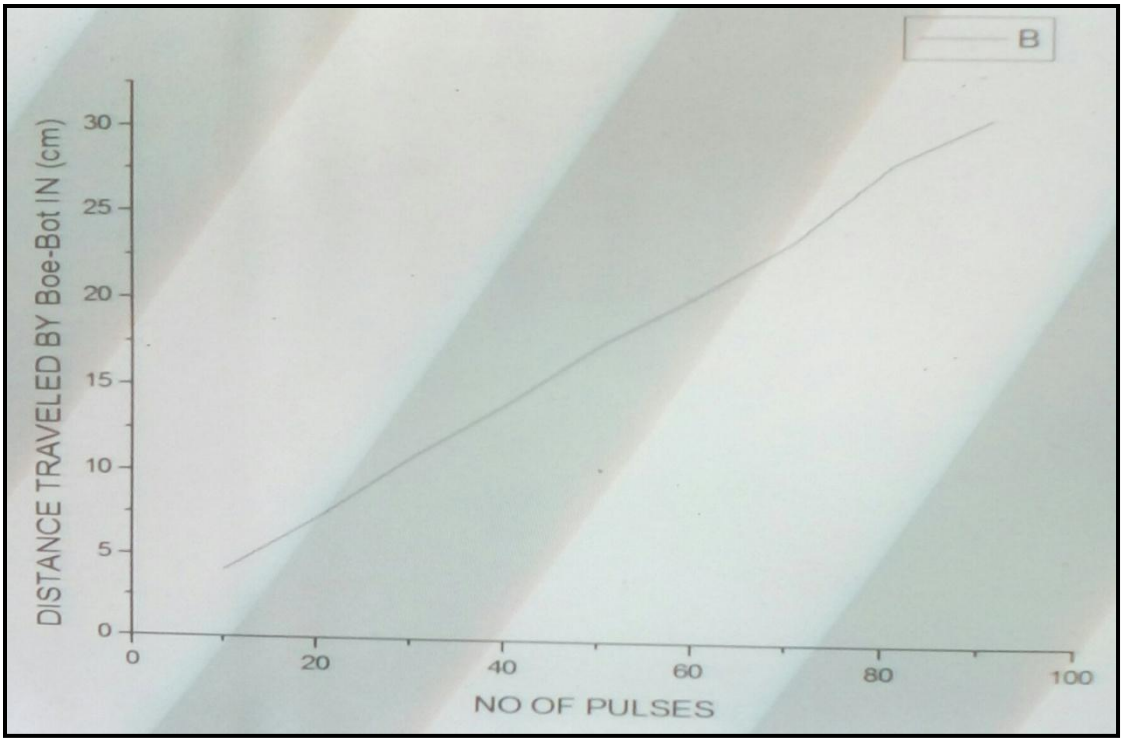

Figure: - 4.9: Graph of distance traversed vs. no of pulses sent

Rotation about the centre of mobile robot (pivoting):

For Boe-Bot‘s counter clockwise rotation i.e. for left turn, Boe-Bot‘s right wheel will have to rotate clockwise and left wheel will have to rotate also clockwise. For right turn or clockwise rotation, both wheels will be reversed. So following commands will be used to perform left turn and right turn operations at maximum speed by rotating the robot about its centre.

For left turn (or counter clockwise rotation)

```
FOR counter = 1 TO n
PULSOUT 15, 650
PULSOUT 14, 650
PAUSE 20
NEXT
```

For right turn (or clockwise rotation)

```
FOR counter = 1 TO n
PULSOUT 15, 850
PULSOUT 14,850
PAUSE 20
NEXT
```

In both cases the value of the variable n (i.e. the number of execution of the loop) will determine the time of running the motors, and hence the angle rotated by the robot about its centre. The angles of rotation at different number of pulses have also been measured at maximum speed, and are shown in table 4.2.

TABLE 4.2: ANGLE OF ROTATION AGAINST NUMBER OF PULSES SENT

OBSERVATION NO.	DURATUON ARGUMENT		NO. OF PULSES	ANGLE OF ROTATION (CCW) (Degree)	ANGLE (CCW) (Brad)
	P15 (left)	P14 (right)			
1	650	650	10	41	29.15
2	650	650	15	55	39.11
3	650	650	20	80	56.89
4	650	650	25	96	68.26
5	650	650	30	122	86.75
6	650	650	35	146	103.82

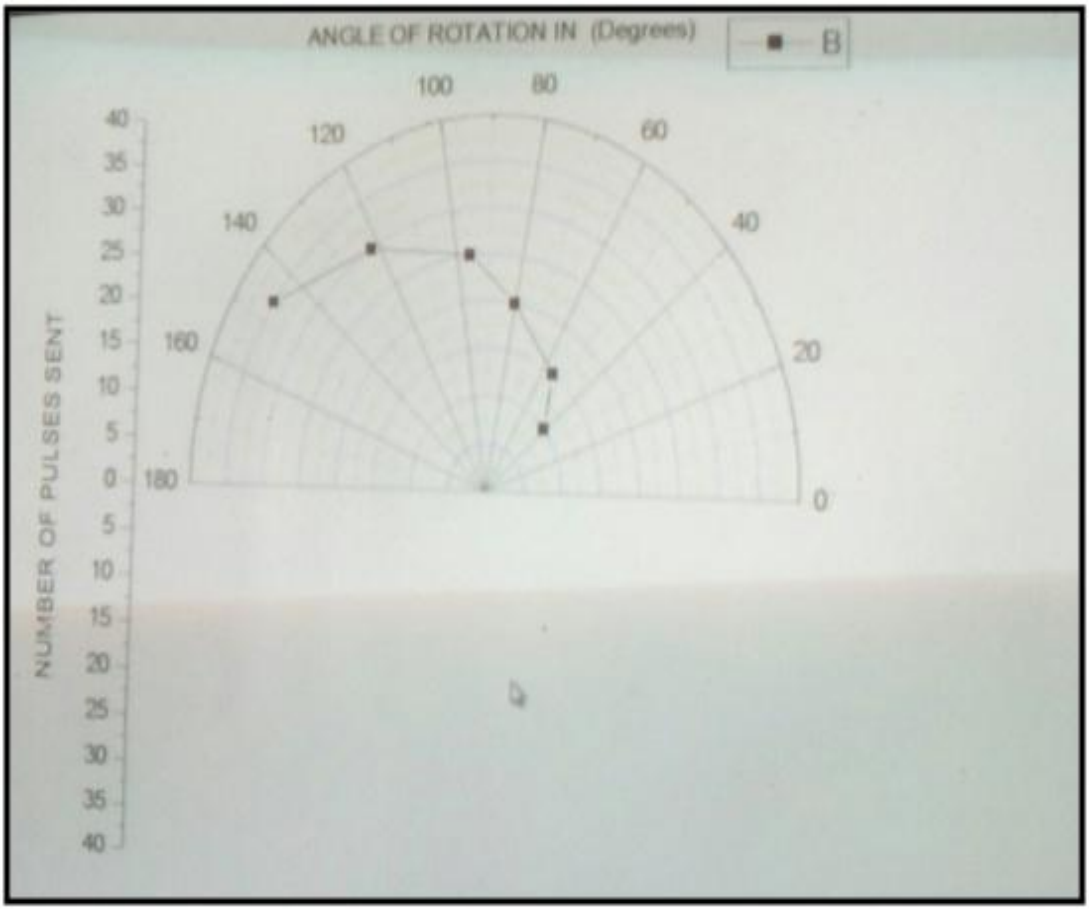

Figure 4.10 Graph of no of pulses sent vs. angle of rotation in CCW direction

4.3 PARALLAX PING ULTRASONIC RANGE SENSOR

The Ping sensor is a device which can be used with the BASIC Stamp to measure how far away an object is. Its range is 3 centimeters to 3.3 meter. It's also remarkably accurate, easily detecting an object's distance down to the half centimeter [26]. The Ping ultrasonic range sensor is shown in fig 4.11.

Figure 4.11 Ping range sensor

The Ping sensor sends a brief chirp with its ultrasonic speaker (transmitter) and makes it possible for the BASIC Stamp to measure the time it takes the echo to return to its ultrasonic microphone (receiver). The BASIC Stamp starts by sending the Ping sensor a pulse to start the measurement. Then the Ping sensor waits long enough for the BASIC Stamp program to start a PULSIN command. At the same time the Ping sensor chirps its 40 kHz tone, it sends a high signal to the BASIC Stamp. When the Ping sensor detects

the echo with its ultrasonic microphone, it changes that high signal back to low. The BASIC Stamp's PULSIN command stores how long the high signal from the Ping sensor lasted in a variable. The measurement is how long it took sound to travel to the object back. The program can calculate the object's distance in centimeter, inch, feet etc. from this time measurement using the speed of sound in air.

Ping ultrasonic range sensor provides an easy method of distance measurement. This sensor is perfect for any number of applications that require performing measurements between moving or stationary objects. It provides precise, non-contact distance measurements within a 3 cm to 3.3 m range.Fig.3.8 shows a schematic diagram for connecting the Ping sensor to the Board of Education carrier board. The Ping sensor has built-in protection against programming mistakes (and wiring mistakes), so there's no need to use any resistor between P4 (Pin 4) and the Ping sensor's SIG terminal.

A pulse to pin 4 (P4) that lasts 10 µs (sent by PBASIC command PULSOUT 4,5) is easily detected by the Ping sensor, and it only takes a small amount of time for the BASIC Stamp to send. A PULSIN command that stores the duration of the Ping sensor's echo pulse has to come immediately after the PULSOUT command. The result the PULSIN command stores is the round trip time for the Ping sensor's chirp to get to the object, reflect and return.

The following commands in PBASIC can be used for this purpose:

```
PULSOUT 4, 5
PULSIN 4, 1, time
Dist = time** 2251
PAUSE 100
```

The echo time measured by the above program can be converted to object's distance in centimeters or other units by multiplying it with a suitable constant using the speed of sound in air.

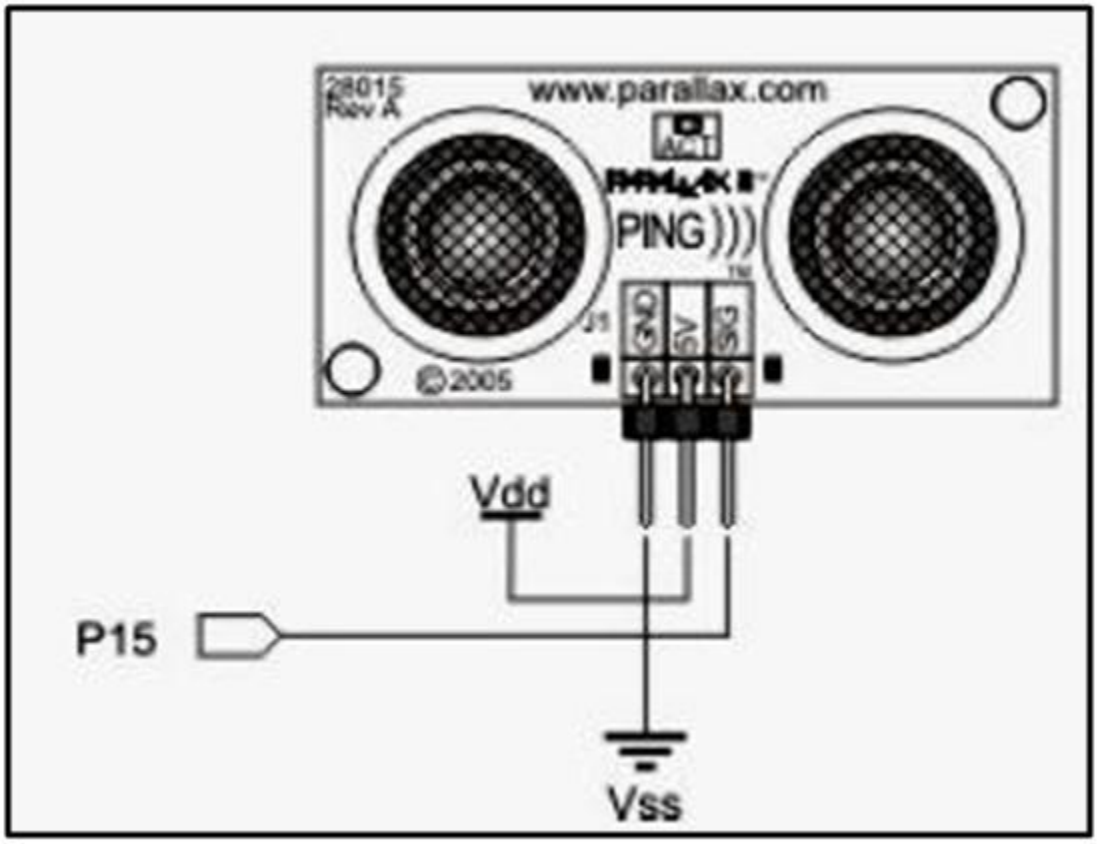

Figure 4.12 Ping sensor schematic and wiring diagram

4.4 PROGRAM DEVELOPMENT FOR PATH PLANNING OF BOE-BOT MOBILE ROBOT WITH ULTRASONIC RANGE SENSOR USING E-BUG ALGORITHM

A program has been developed in PBASIC language in the present project for path planning of the Boe-Bot mobile robot using E-Bug algorithm with the Ping ultrasonic range sensor mounted on the system carrier board. The PBASIC program has been developed to instruct the mobile robot to navigate from a start point to a goal point in presence of obstacles. The start point has been assumed to be the origin of a co-ordinate system, and the co-ordinates of the goal point are entered as variables during execution of the program. The Boe-Bot mobile robot is assumed to be parallel to the x-axis at the beginning. The mobile robot rotates towards the goal point and starts moving towards it until any obstacle is detected on its path by the range sensor, whose output is monitored continuously during movement of the robot. Only the nearest obstacle, which is within a specified range (taken as 20 cm), has been considered. If an obstacle is detected, the robot rotates both left and right searching for 'sudden point', where the reading of the range sensor changes sharply. A change in the value from 20 cm (the distance at which an obstacle is detected) to 25 cm or greater in the reading of the range sensor output has been considered for detecting 'sudden point'. At the same time the robot determines the coordinate of the sudden point by mathematical operation. The robot then moves towards that 'sudden point' out of the two 'sudden points' for an obstacle, for which the sum of the distance between the current position of the Boe-Bot to sudden point and sudden point to goal or target position is minimum. The distance between the current point to sudden point has been assumed as 25 cm (since it detects a sudden point having distance not less than 25 cm), and the distance of sudden point to goal has been calculated using trigonometry and properties of triangle. When the robot moves towards the 'sudden point', it also checks for the presence of any other obstacle on its path by monitoring continuously the output of the range sensor. If another obstacle is detected, the process is repeated, otherwise the robot moves through a further distance (taken as 5 cm) past the 'sudden point' to prevent any collision with the obstacle when it again rotates towards the goal point and moves straight towards it. The process is repeated until

the mobile robot reaches the goal point, which is assured by considering the distance of the current robot position from the goal point.

Now the ultrasonic range sensor detects obstacle within a wide angle from normal direction. This angular spread has been made narrow by attaching two papers in the form of hollow cylinders to both the ultrasonic transmitter and receiver. Still it detects object with in a small angular spread from normal direction.

The mobile robot has been assumed to be a point in the algorithm. Since the ultrasonic range sensor detects obstacle within some angular spread from normal direction, the 'sudden point' is detected at a small distance from the obstacle so that when the mobile robot moves towards 'sudden point', it automatically moves away by a small distance from the obstacle avoiding any collision. As stated above, the robot is also forced to move past the 'sudden point' by a small distance before turning towards goal point.

The complete PBASIC program developed for the path planning of Boe-Bot mobile robot having ultrasonic range sensor has been listed in Table 4.3

Table 4.3: PBASIC PROGRAM DEVELOPED FOR PATH PLANNING USING E-BUG ALGORITHM

```
{$STAMP BS2}
{$PBASIC 2.5}
x1 VAR Word
x2 VAR Word
y1 VAR Word
y2 VAR Word
c VAR Word
s VAR Word
n VAR Byte
m VAR Byte
ir VAR Byte
```

```
il VAR Byte
i VAR Byte
time VAR Word
d VAR Byte
dist VAR Byte
ur VAR Byte
ul VAR Byte
ang VAR Byte
angr VAR Word
angr=0
DEBUG "Enter start point (x1,y1)"
DEBUGIN SDEC x1
DEBUGIN SDEC y1
DEBUG "Enter goal point (x2,y2)"
DEBUGIN SDEC x2
DEBUGIN SDEC y2
d=SQR((x2-x1)*(x2-x1)+(y2-y1)*(y2-y1))

DO WHILE (d>1)
m=d*65/22
ang=(x2-x1) ATN (y2-y1)
angr = ang - angr
IF (angr.BIT15=1) THEN
angr=angr + 256
ENDIF

IF (angr=0) THEN
GOTO L0
ENDIF
```

```
'Rotate towards goal
IF angr<=128 THEN
n=angr*20/64
FOR i=1 TO n
GOSUB ccw
NEXT
ELSE
n=(256-angr)*20/64
FOR i=1 TO n
GOSUB cw
NEXT
ENDIF

L0:

‘Move towards goal & detects obstacle
i=0

L1:

PULSOUT 4, 5
PULSIN 4,1, time
dist = time**2251
IF (dist>20 AND i<=m) THEN
GOSUB fm
i=i+1
GOTO L1
ENDIF
```

```
'Calculate ( x,y ) for new position
c=COS(ang)
IF (c.BIT15=1) THEN
x1=x1-((i*22/65)*ABS(c)/127)
ELSE
x1=x1+((i*22/65)*ABS(c)/127)
ENDIF

s=SIN(ang)
IF (s.BIT15=1) THEN
y1=y1-((i*22/65)*ABS(s)/127)
ELSE
y1=y1+((i*22/65)*ABS(s)/127)
ENDIF

d =SQR((x2-x1)*(x2-x1) + (y2-y1)*(y2-y1))
IF (i>m) THEN
GOTO L3

ENDIF

'Obstacle detected & search sudden points on both sides

ir=0

DO WHILE (dist<25)
GOSUB cw
PULSOUT 4, 5
PULSIN  4,1, time
dist = time**2251
ir=ir+1
LOOP
ur = 25 + SQR((d*d) + (25*25) - (2*d*25*(COS(ir*64/20))/127))
```

```
‘Rotate to original orientation
FOR i = 1 TO ir
GOSUB ccw
NEXT
dist=20
il=0

DO WHILE (dist<25)
GOSUB ccw
PULSOUT 4, 5
PULSIN 4,1, time
dist = time**2251
il=il+1
LOOP
ul = 25 + SQR((d*d) + (25*25) - (2*d*25*(COS(il*64/20))/127))

‘Rotate to original orientation
FOR i = 1 TO il
GOSUB cw
NEXT

'Rotate towards sudden points for minimum distance towards goal
IF (ur<ul) THEN
n=ir
FOR i=1 TO n
GOSUB cw
NEXT
ang = ang - (n*64/20)
ELSE
n=il
FOR i=1 TO n
GOSUB ccw
NEXT
```

```
ang = ang + (n*64/20)
ENDIF

‘Move towards sudden point for minimum distance towards goal
m = 25*65/22
FOR  i = 1 TO m
GOSUB fm
NEXT

'Calculate (x,y) for new  position
c=COS(ang)
IF (c.BIT15=1) THEN
x1=x1-((i*22/65)*ABS(c)/127)
ELSE
x1=x1+((i*22/65)*ABS(c)/127)
ENDIF

s=SIN(ang)
IF (s.BIT15=1) THEN
y1=y1- ((i*22/65)*ABS(s)/127)
ELSE
y1=y1+ ((i*22/65)*ABS(s)/127)
ENDIF

angr=ang
L3:
d=SQR(((x2-x1)*(x2-x1))+((y2-y1)*(y2-y1)))
LOOP
DEBUG "Reached goal"
END
```

```
ccw:
'Subroutine for ccw rotation
PULSOUT 15,650
PULSOUT 14,650
PAUSE 20
RETURN

cw:
'Subroutine for cw rotation
PULSOUT 15,850
PULSOUT 14,850
PAUSE 20
RETURN

fm:
'Subroutine for forward movements
PULSOUT 15,850
PULSOUT 14,650
PAUSE 20
RETURN
```

4.5 EXPERIMENTATION, RESULTS AND DISCUSSIONS

The developed program has been run successfully for different layouts of obstacles and goal points with different orientations of obstacles. The photographic view of one such layout of workspace with the robot at the initial position and orientation has been shown in figure 4.13.

According to the developed program, the robot moves from its start point to goal point with the help of some subroutine and commands used in PBASIC language. At first the Boe-Bot is facing towards the X-axis then it is rotated towards goal. Then the robot moves towards goal until it finds any obstacle with the help of ultrasonic range sensor. When the sensor detects obstacle then the program starts finding sudden points by rotating the robot on both sides with the help of range sensor. Then the program determines the sum of sub paths from robot's current position to sudden point and sudden point to goal point on both sides. Whichever sum of sub paths is minimum the robot moves towards corresponding sudden point with the help of forward movement subroutine.

On reaching the sudden point, the robot again rotates towards the goal. The process is repeated until the goal point is reached. In this way the robot moves from start to goal point in a near optimal path.

Photographic views of different position along the path are shown in the figure 4.14 and 4.15

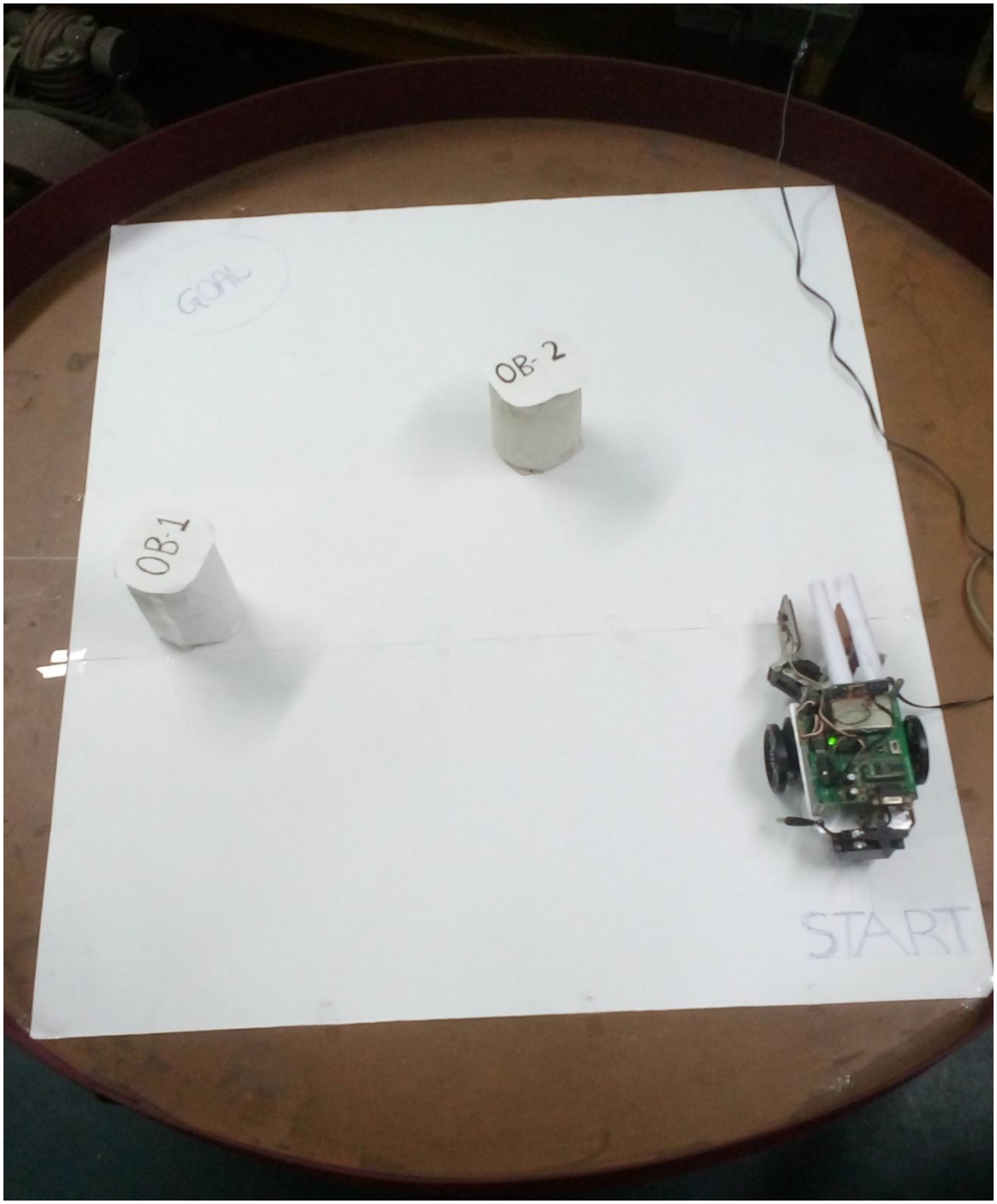

Figure 4.13 Photographic view of one layout of workspace consisting of Boe-Bot mobile robot and obstacles with the robot in the initial position and orientation

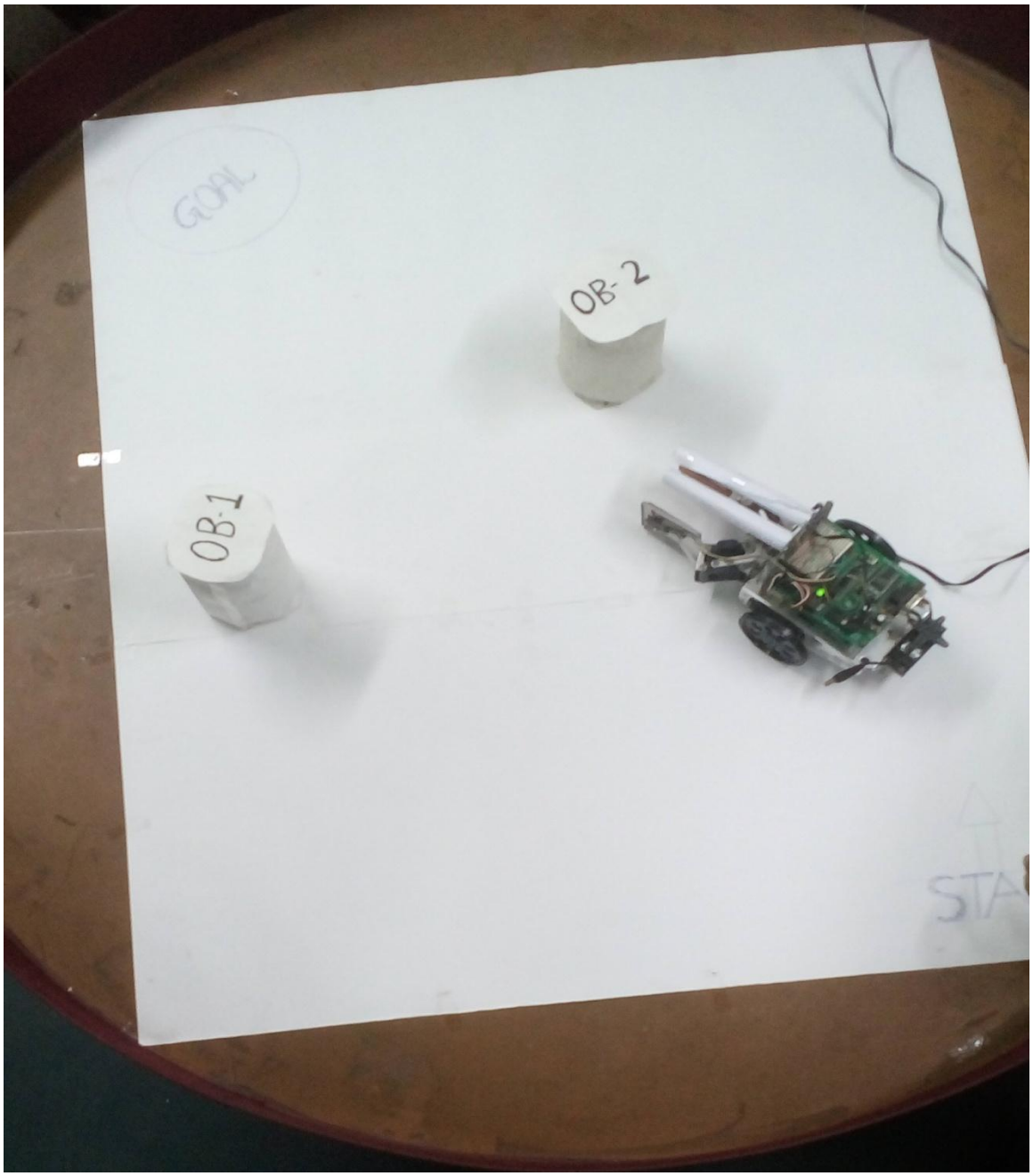

Figure 4.14 Photographic view of the position of Boe-Bot mobile robot when an obstacle is detected

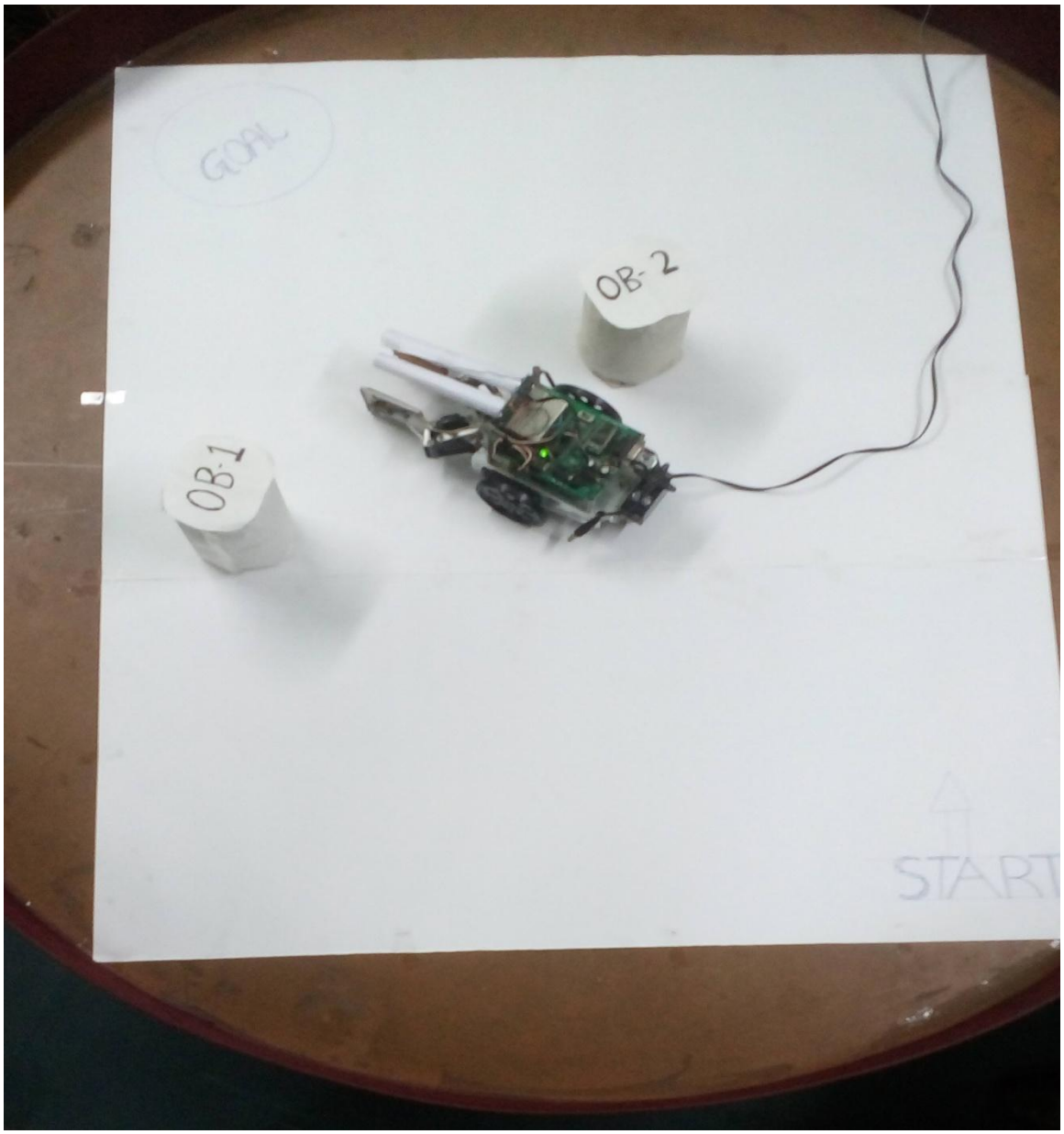

Figure 4.15 Photographic view of the position of Boe-Bot mobile robot when it reaches a sudden point after an obstacle is detected.

Chapter-5

CONCLUSIONS

Based on the foregoing analysis, study of the algorithm and its program development, experimentations and results on 'Online Path Planning of a Mobile Robot with an Ultrasonic Range Sensor Using E-Bug Algorithm' for the present project, the following general conclusions may be drawn:

1) Various path planning techniques and algorithms including Bug family of algorithms have been studied thoroughly, and the PointBug algorithm has been found suitable in many cases.

2) As the PointBug algorithm possesses some major limitations or deficiencies in determining the near optimal path, some improved path planning algorithms overcoming these limitations have also been studied.

3) E-Bug Algorithm, which has been used in the present project, is an improved algorithm, and is based on the PointBug algorithm.

4) In E-Bug Algorithm, the next point to move towards a goal point is determined by the output of an ultrasonic range sensor. The sudden change in distance from the sensor to the nearest obstacle is detected from the output of the range sensor by suitably defining a well marked difference value in the readings of the sensor. In E-Bug algorithm the sum of the distances of sub paths from current position to sudden point and sudden point to target or goal point is determined for obtaining the near optimal path and at the same time avoids forming endless cycle.

5) An arrangement has been set up for a workspace consisting of a Parallax Boe-Bot mobile robot with a Parallax Ping ultrasonic range sensor and multiple obstacles on a suitable work-table in the Robotics Laboratory of Production Engineering Department.

6) The Ping ultrasonic sensor has been suitably mounted on a breadboard fixed on the Boe-Bot mobile robot system, and necessary hardware connection has been made to connect it to the BASIC Stamp microcontroller which runs the mobile robot.

7) A program based on modified PointBug Algorithm named as E-Bug has been developed in PBASIC language for producing necessary movements of the Boe-Bot mobile robot in presence of static obstacles for moving from a start point to a goal point using the output of the range sensor.

8) The size of the robot (assumed to be a point in the algorithm) has been taken care of by the fact that the 'sudden point' is detected at some distance from the obstacle (25 cm), since the range sensor detects obstacle within some angular spread from normal distance so that when the mobile robot moves towards 'sudden point', it automatically moves away by a small distance from the obstacle avoiding any collision. The robot is also forced to move past the 'sudden point' by a small distance before turning towards goal point.

9) The PBASIC program has been run successfully for different layouts of workspace.

Further scope of the present project includes development of path planning algorithms in presence of static as well as dynamic (moving) obstacles in the environment using camera or other important vision sensors.

Chapter-6

REFFERENCES

[1] James Ng. “An Analysis of Mobile Robot Navigation Algorithms in Unknown Environments” .Ph.D. thesis, School of Electrical Electronic and Computer Engineering, The University of Western Australia. Feb. 2010.

[2] Lumelsky V., T. Skewis, “Incorporating Range Sensing in the Robot Navigation Function Transactions On System, Man, And Cybernetics”, 1990. 20(5): p. 12.

[3] Khatib, “Real-Time Obstacle Avoidance for Manipulators and Mobile Robots”, in International Journal of Robotics Research: p. 90- 98, 1986.

[4] Al-Taharwa I., Sheta A., Al-Weshah “A Mobile Robot Path Planning Using Genetic Algorithm in Static Environment”, Journal of Computer Science, 4 (4): 341-344, 2008.

[5] Vladimir J. Lumelsky and Alexander A. Stepanov: “Path- Planning Stategies for a Point Mobile Automaton Moving Amidst Unknown Obstacles of Arbitrary Shape”. Algorithmica (1987) 2: pages 403-430

[6] Sankaranarayanan A., M. Vidyasagar, “Path Planning for Moving a Point Object Amidst Unknown Obstacles in a Plane”, the universal lower bound on the worst path lengths and a classification of algorithms, IEEE International Conference on Robotics and Automation, 1991: p. 1734 - 1741.

[7] Kamon I.E. Rivlin, “Sensory-Based Motion Planning with Global Proofs”, IEEE Transactions on Robotics and Automation, 1997, 13 (6).

[8] Kamon I., E. Rimon, E. Rivlin, “TangentBug A Range-Sensor-Based Navigation Algorithm”, the Int. Journal of Robotics Research, 1998. 17 (9) : p. 934-953.

[9] Noborio H., K. Fhjimura, Y. Horiuchi, “A Comparative Study of Sensor Based Path Planning Algorithms in an Unknown Maze”, in Int. Conf. on intelligent Robots and Systems, 2000, Japan, IEEE, RSJ.

[10] Bella, J.E, P. R. Mc Mullenb, “Ant Colony Optimization Techniques for the Vehicle Routing Problem”, Advanced Engineering Informatics, 2004. 18: p. 41-48.

[11] N. Bin, C.X, Z. Liming, X. Wendong, " Recurrent Neural Network for Robot Path Planning Parallel and Distributed Computing Applications and Technologies", 2004, 3320/2005: (Springer Berlin / Heidelberg).

[12] Xiao-Guang, Xiao-Wei Fu, Da-Qing Chen, "Genetic-Algorithm-Based Approach to UAV Path Planning Problem" in 5th WSEAS Int. Conf. on SIMULATION, MODELING AND OPTIMIZATION", 2005. Corfu, Greece.

[13]J. Chestnutt and M.Lau," Footstep planning for the Honda ASIMO Humanoid ", IEEE international conference on robotics and Automation, pages 629-634, Apr 2005.

[14] Santiago Garrido, Luis Moreno, Mohamed Abderrahim, Fernando Martin, "Path Planning for Mobile Robot Navigation using Voronoi Diagram and Fast Marching", in International Conference on Intelligent Robots and Systems Beijing, China, 2006. And "Robotics Lab., Carlos III Univ., Madrid DOI: 10.1109/ROBOT.2006.1642165 ,Conference: Proceedings of the 2006 IEEE International Conference on Robotics and Automation, ICRA 2006, May 15-19, 2006, Orlando, Florida, USA".

[15] Ng J., T. Bräunl, "Performance Comparison of Bug Navigation Algorithms", Journal of Intelligent and Robotic Systems, 2007, p: 73- 84.

[16] Y. Wang, P. sillitoe, J. Mulvancy, "Mobile Robot Path Planning in Dynamic Environments", IEEE International Conference on Robotics and Automation, pages 71-76, May,2007.

[17]Jasmin Velagic, Bakir Lacevic and Nedim Osmic(2008).Nonlinear motion Control of Mobile Robot Dynamic Model, Motion Planning, Xing-Jian Jing (Ed.), ISBN: 978-953-7619-01-5, InTech, DOI: 10.5772/5997. Available from: http://www.intechopen.com/books/motion planning/nonlinear motion control of mobile robot dynamic model.

[18] Al-Taharwa I., Sheta A., Al-Weshah "A Mobile Robot Path Planning Using Genetic Algorithm in Static Environment", Journal of Computer Science, 4 (4): 341-344, 2008.

[19]Priyadarshi Bhattacharya,Marina L.Gavrilova, “Roadmap Based Path Planning Using the Voronoi Diagram for a Clearance-Based Shortest Path”, in IEEE Robotics & Automation Magazine, JUNE 2008.

[20] M. Tarokh, “Hybrid Intelligence Path Planning for Articulated Rovers in Rough Terrian”, Fuzzy Sets and Systems, Vol 159, Pages 2927-2937, Nov 2008.

[21] L. Hu, Z. Gu, “ Research and Realization of Optimum Route Planning in Vehicle Navigation Systems based on a Hybrid Genetic Algorithm”, Proceedings of the Institution of Mechanical Engineering Part-D Journal of Automobile Engineering, Vol 22, Pages 757-763, May 2008.

[22] Kumar, E.V, M. Aneja, D. Deodhare, “Solving a Path Planning Problem in a Partially Known Environment using a Swarm Algorithm”, in IEEE International Symposium on Measurements and Control in Robotics. 2008. Bangalore, India.

[23] Santiago Garrido, Luis Moreno, M. Abderrahim, D. Blanco, “Robot Navigation using Tube Skeletons and Fast Marching”, Advanced Robotics, ICAR 2009.

[24] Susnea I., Viore Minzu, Grigore Vasiliu, “Simple Real-Time Obstacle Avoidance Algorithm for Mobile Robots”, in 8th WSEAS International Conference on Computational intelligence, man-machine systems and cybernetics (CIMMACS'09) 2009.And www.wseas.us/e-library/conferences/2009/tenerife/CIMMACS/CIMMACS-03.

[25] International journal on computer application in technology, volume 44 issue 4, October 2012, pages 303-311.

[26]DOI: 10.1109/LARC.2011.6086816 Conference: IEEE IX Latin American Robotics Symposium and IEEE Colombian Conference on Automatic Control, LARC 2011, Bogota, Colombia, October 1-4, 2011.

[27] Buniyamin N., Wan Ngah, W.A.J., Sariff N., Mohamad Z. “A Simple Local Path Planning Algorithm for Autonomous Mobile Robots”, in International Journal Of Systems Applications, Engineering & Development Issue 2, Volume 5, 2011.

[28] Procedia Engineering 41 (2012) 182–188, www.elsevier.com/locate/procedia.

[29]Aditya Mahadevan, Nancy M.,"A Sampling-Based Approach to Probabilistic Pursuit Evasion", in IEEE International Conference on Robotics and Automation River Centre, Saint Paul, Minnesota, USA, 2012.

[30] Available at: http://www.sciencedirect.com, information sciences 280 (2014) 64-81, Journal homepage: www.elsevier.com/locate/ins.

[31] Available at: http://www.sciencedirect.com, Applied Soft Computing 30 (2015) 319-328 Journal homepage: www.elsevier.com/locate/asoc.

[32] Available at: http://www.sciencedirect.com, Expert Systems with Applications 42 (2015) 5177–5191, journal homepage: www.elsevier.com/locate/eswa.

[33] Available at: http://www.sciencedirect.com, Engineering Applications of Artificial Intelligence 44 (2015) 123–136, journal homepage: www.elsevier.com/locate/engappai.

[34] Available at: http://www.sciencedirect.com, Computational Geometry: Theory and Applications, www.elsevier.com/locate/comgeo.

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

[36] Schwartz, J., M. Sharir, On the piano mover's problem II: General techniques for computing topological properties of real algebraic manifolds. Adv. Appl. Math.,, 1983. 4: p. 298-351.

[37] Sedighi, K., Ashenay H.,Manikas K. , Wainwright T. W., Tai R. L. H.- M. Autonomous local path planning for a mobile robot using a genetic algorithm.in Congress on Evolutionary Computation (CEC2004). 2004: IEEEXplore.

[38] Farouk Meddah and Lynda Dib: "P*: A new path planning algorithm for autonomous robot in an unknown environment".Second International Conference on Advances in Computing, Electronics and Communication - ACEC 2014. Zurich, Switzerland. Pages 42-46

[39] Meddah, F. and Bid, L. , E-Bug: New Bug Path-planning algorithm for autonomous Robot in unknown environment Available at: https://www.researchgate.net/publication.

[40] Ricardo A. Langer, Leandro S. Coelho Gustavo H. C. Oliveira.et al. “K-Bug, a New Bug Approach for Mobile Robot's Path Planning”. 16th IEEE International Conference on Control Applications, Part of IEEE Multi-conference on Systems and Control. Singapore, Oct. 2007. Pages 403-408

[41] BASIC Stamp Syntax and Reference Manual 2.2. www.parallax.com and PARALLAX INC. Robotics with the Boe-Bot student guide, version 2.2