Development of an Autonomous Mobile Robot with Planning and Location in a Structured Environment

Lucas Tiago Eckert

Dissertation presented to the School of Technology and Management of Bragança to obtain the Master Degree in Engenharia Industrial.

Work oriented by:

Professor PhD José Luis Sousa Magalhães Lima

Professor PhD Paulo Gomes Costa

Professor PhD Alberto Yoshihiro Nakano

Bragança

2019

Development of an Autonomous Mobile Robot with Planning and Location in a Structured Environment

Lucas Tiago Eckert

Dissertation presented to the School of Technology and Management of Bragança to obtain the Master Degree in Engenharia Industrial.

Work oriented by:

Professor PhD José Luis Sousa Magalhães Lima

Professor PhD Paulo Gomes Costa

Professor PhD Alberto Yoshihiro Nakano

Bragança

2019

Acknowledgement

Firstly, I would like to express my sincere gratitude to my advisor Prof. PhD José Luis Lima for the continuous support of my Master study and related research, for his patience, motivation, and immense knowledge. His guidance helped me in all the time of research and writing of this thesis.

To Professor PhD Alberto Nakano, co-supervisor of this work, my gratitude for your availability, encouragement, immense knowledge, and support in the preparation of this work.

To Professor PhD Paulo Costa, co-supervisor of this work, my gratitude for the guidance and willingness to share his vast knowledge in the area.

To Engineer Luis Piardi, my gratitude for your availability, immense knowledge, and support during the development of this work.

To all the friends and colleagues who, directly or indirectly, contributed to the preparation of this study, my gratitude for the patience, attention and strength you have given in difficult moments.

Finally, I must express my very profound gratitude to my family for providing me with unfailing support and continuous encouragement throughout my years of study and through the process of researching and writing this thesis. This accomplishment would not have been possible without them. Thank you.

Abstract

With the advance of technology mobile robots have been increasingly applied in the industry, performing repetitive work with high performance, and in environments that pose risks to human health. The present work plans and develops a mobile robot platform for the micromouse competition. The micromouse consists of a small autonomous mobile robot that, when placed in an unknown labyrinth, is able to map it, search for the best path between the starting point and the goal and travel it in the shortest possible time.

To accomplish these tasks, the robot must be able to self-locate, map the maze as it traverses it and plan paths based on the map obtained. The developed self-localization method is based on the odometry, the laser sensors present in the robot and on a previous knowledge of the start point and the configuration of the environment. Several methodologies of locomotion in unknown environment and route planning are analyzed in order to obtain the combination with the best performance.

In order to verify the results, the present work is developed in real environment, in 3D simulation and also with a hardware in the loop capability. Labyrinths from previous competitions are used as basis for comparing methodologies and validating results. At the end it presents the algorithm capable of fulfilling all the requirements of the micromouse competition together with the results of its evaluation run.

Keywords: Micromouse; Autonomous Mobile Robot; Path Planning; Mapping; Self-Location.

Resumo

Com o avanço da tecnologia, os robôs móveis têm sido cada vez mais aplicados na indústria, realizando trabalhos repetitivos com alto desempenho e em ambientes que expõem riscos à saúde humana. O presente trabalho planeja e desenvolve um robô móvel para a competição micromouse. O micromouse consiste em um pequeno robô autônomo que, ao ser colocado em um labirinto desconhecido, é capaz de mapeá-lo, procurar o melhor caminho entre o ponto de partida e o objetivo, e percorrê-lo no menor tempo possível.

Para realizar estas tarefas, o robô deve ser capaz de se auto-localizar, mapear o labirinto enquanto o percorre e planejar caminhos com base no mapa obtido. O método de auto-localização desenvolvido baseia-se na odometria, nos sensores a laser presentes no robô e em um prévio conhecimento do ponto de início e da configuração do ambiente. Diversas metodologias de locomoção em ambiente desconhecido e planejamento de rotas são analisadas buscando-se obter a combinação com o melhor desempenho.

Para averiguação de resultados o presente trabalho desenvolve-se em ambiente real e em simulação 3D com *hardware in the loop*. Labirintos de competições anteriores são utilizados de base para o comparativo de metodologias e validação de resultados. Ao final apresenta-se o algoritmo capaz de cumprir todas as exigências da competição micromouse juntamente com os resultados em sua corrida de avaliação.

Palavras-chave: Micromouse; Robô Móvel Autonomo; Planejamento de Caminho; Mapeamento; Auto Localização.

Contents

Acknowledgement **v**

Abstract **vii**

Resumo **ix**

Acronyms **xxi**

1 Introduction **1**

1.1 Motivation 1

1.2 Objectives 3

1.3 Dissertation Outline 4

2 Related Work **7**

2.1 Robotics Competitions 7

2.1.1 RoboCup Soccer Competition 8

2.1.2 Urban Search and Rescue Competitions 9

2.1.3 Robot@Factory 9

2.1.4 DARPA Grand Challenge 10

2.2 Mobile Robots Applications 11

2.3 Locomotion of Wheeled Mobile Robot 12

2.3.1 Ackerman Steering Geometry 13

2.3.2 Omnidirectional Geometry 13

2.3.3 Differential Geometry 14
2.4 World Representation 14
2.4.1 Topological 15
2.4.2 Grid-based 16
2.5 Simultaneous Localization and Mapping 17
2.6 Path Planning 18
2.6.1 Cell Decomposition 18
2.6.1.1 Approximate Cell Decomposition 19
2.6.1.2 Adaptive Cell Decomposition 19
2.6.1.3 Exact Cell Decomposition 21
2.6.2 Roadmap Methods 21
2.6.2.1 Visibility Graphs 21
2.6.2.2 Voronoi Diagrams 22
2.6.2.3 Probabilistic Roadmaps 23
2.6.3 Potential Fields 24
2.7 Trajectory Planning 25
2.7.1 A* Algorithm 25
2.7.2 Dijkstra's Algorithm and Best-first Search Algorithm 27

3 Real Robot Description **29**
3.1 Structural Constitution of the Robot 29
3.2 Kinematic Model of Differential Robot 32
3.2.1 Pose 32
3.2.2 Linear and Angular Velocity 33
3.3 Electronic Hardware 34
3.3.1 WEMOS D1 Mini 35
3.3.2 Motor Shield 35
3.3.3 Battery Shield 36
3.3.4 Step-up and motor 36

3.3.5 Sensors . . . 37
3.3.5.1 Odometer . . . 38
3.3.5.2 Laser Distance Sensor . . . 40
3.4 Robot Software . . . 41

4 Simulation Model 43
4.1 SimTwo Simulator . . . 43
4.1.1 Maze Generator . . . 44
4.1.2 Robot Configuration . . . 46
4.1.3 Robot Control . . . 47
4.1.4 Results Presentation . . . 48
4.2 Hardware in the Loop . . . 49

5 Localization, Mapping, Path planning and Control 53
5.1 Maze representation . . . 53
5.2 Localization . . . 54
5.3 Control . . . 57
5.3.1 Centralization Algorithm . . . 58
5.3.1.1 Both Close Lateral Walls or a Large Frontal Wall . . . 58
5.3.1.2 One of the Close Lateral Walls . . . 59
5.3.1.3 One of the Sidewalls a Cell Away . . . 59
5.3.1.4 Other cases . . . 61
5.3.2 Curves . . . 61
5.4 Search Algorithms . . . 62
5.4.1 A* Modified . . . 62
5.5 Mapping . . . 63
5.6 Path Planning and Evaluation Run . . . 65

6 Results 67
6.1 Simulation . . . 67

6.1.1 Localization 68
6.1.2 Mapping 69
6.1.2.1 Wall Follower 70
6.1.2.2 Wall Follower with repositioning algorithm 71
6.1.3 Path Planning 73
6.1.3.1 2008 All Japan Micromouse Expert Contest 73
6.1.3.2 2018 APEC Micromouse Contest 75
6.1.3.3 2018 UK Micromouse 76
6.1.3.4 Comparison 77
6.1.4 Evaluation Run 77
6.2 Real Robot 78

7 Conclusions and Future Work 81

7.1 Developed Work 81
7.2 Future Works 82

A Publications 92

A.0.1 Paper Submitted and Accepted 92
A.0.2 Papers Submitted and Awaiting Evaluation 93

List of Tables

3.1 Dimensions of the micromouse robot 31
3.2 Micro metal gear motor specifications 37

4.1 Micromouse simulated structure. 46
4.2 Position of the sensors in Simtwo simulation. 47

6.1 Results of the wall follower algorithm. 72
6.2 Results of the wall follower with repositioning algorithm. 73
6.3 Search Algorithms applied on 2008 All Japan Micromouse results 74
6.4 Search Algorithms applied on 2018 APEC Micromouse results 75
6.5 Search Algorithms applied on 2018 UK Micromouse results 76
6.6 Evaluation run results. 78

List of Figures

1.1 APEC 2017 Micromouse Contest Venue [5]. 2

2.1 RoboCup soccer competitions : (a) Simulation league. (b) Middle size league. (c) Humanoid league. 8
2.2 A USAR robot operating in the National Institute of Standards and Technology (NIST) arena [13]. 9
2.3 Robot@Factory competition arena [16]. 10
2.4 Defense Advanced Research Projects Agency (DARPA) Gran Challenge. Adapted from [17]. 11
2.5 Curiosity - Robot for exploration of the planet Mars (NASA). Adapted from [21]. 12
2.6 Ackerman steering geometry realizing a turn. 13
2.7 Robot with omnidirectional geometry (Uranus) [20]. 14
2.8 Robot with differential geometry (Cyo, produced by Probotics, Inc). 15
2.9 Topological map representation [33]. 15
2.10 Grid-based map representation [33]. 16
2.11 Cell decomposition with different resolutions [43]. 20
2.12 Quadtree representation [33]. 20
2.13 Complete visibility graph [47]. 22
2.14 Voronoi diagram [33] 22
2.15 Probabilistic roadmap [33] 23
2.16 Potential field algorithm [52]. 24

2.17 A Star Algorithm 26

3.1 Assembled micromouse robot : (a) Frontal view. (b) Side view. (c) Top view. 30
3.2 Base plate designed for the micromouse robot : (a) Frontal view. (b) Side view. (c) Top view. 31
3.3 Robot in the plane representing the global robot reference frame. Extracted from [19]. 32
3.4 Micromouse electronic architecture. 34
3.5 WEMOS D1 Mini. 35
3.6 Motor shield. 36
3.7 Battery shield. 37
3.8 Simplified scheme of the process of interaction of the sensors with the environment. 38
3.9 Optical encoder representation. Adapted from [66]. 39
3.10 VL53L0XV2 sensor. 40
3.11 Time-of-flight technology model. Adapted from [70]. 40

4.1 SimTwo micromouse simulation interface. 44
4.2 Maze conversion process. 45
4.3 Simulated micromouse robot : (a) Lateral view. (b) Top view. 47
4.4 Maze presentation in the Spreedsheet window. 49
4.5 Correlation of price, time, risk and precision between simulation and real systems. (adapted from [76]). 50
4.6 Architecture HIL for Micromouse SimTwo simulator. 51
4.7 SimTwo HIL configuration. 52

5.1 Partial representation of the maze matrix (white - free cell, grey - obstacle). 54
5.2 Partial representation of the maze and its axes. 55
5.3 Environment situations: (a) Both close lateral walls. (b) Large frontal wall. 59

5.4 A single wall close. 60
5.5 A single wall a Cell Away. 60
5.6 Ordering a new cell in a binary heap. 62
5.7 Mapping process: (a)Robot's current position. (b) Generated map. 65

6.1 Traveled route. 68
6.2 Comparative between odometry and real robot movimentation. 69
6.3 Maps of past micromouse competitions : (a) 2008 All Japan Micromouse Expert Contest (b) 2018 APEC Micromouse Contest. (c) 2018 UK Micromouse Finals Maze. 70
6.4 Path traversed using the wall-follower algorithm. 71
6.5 Maps generated by the wall follower algorithm 71
6.6 Path traversed using the wall-follower with repositioning algorithm. 72
6.7 Maps generated by the wall follower with repositioning algorithm 72
6.8 Search Algorithms applied on 2008 All Japan Micromouse: (a) Dijkstra's Algorithm (b) Best-first Search (c) A*. (d) A* Modified w/ K=1. (e) A* Modified w/ K=1.5. (f) A* Modified w/ K=2. 74
6.9 Search Algorithms applied on 2018 APEC Micromouse: (a) Dijkstra's Algorithm (b) Best-first Search (c) A*. (d) A* Modified w/ K=1. (e) A* Modified w/ K=1.5. (f) A* Modified w/ K=2. 75
6.10 Search Algorithms applied on 2018 UK Micromouse : (a) Dijkstra's Algorithm (b) Best-first Search (c) A*. (d) A* Modified w/ K=1. (e) A* Modified w/ K=1.5. (f) A* Modified w/ K=2. 76
6.11 Trajectories performed on the evaluation run in the mazes : (a) 2008 All Japan Micromouse Expert Contest (b) 2018 APEC Micromouse Contest. (c) 2018 UK Micromouse Finals Maze. 77
6.12 Assembled maze. 78
6.13 Real maze: (a) Map. (b) Path planning. 79
6.14 Trajectory performed in the evaluation run. 79

Acronyms

AAAI Association for the Advancement of Artificial Intelligence.

AI Artificial Inteligence.

AUVSI Association for Unmanned Vehicle Systems International.

DARPA Defense Advanced Research Projects Agency.

HIL Hardware in the Loop.

IEEE Institute of Electrical and Electronics Engineers.

MDF Medium-density fibreboard.

NIST National Institute of Standards and Technology.

PWM Pulse Width Modulation.

SLAM Simultaneous Localization and Mapping.

UAV Unmanned aerial vehicle.

USAR Urban Search and Rescue.

Chapter 1

Introduction

This work deals with the study and development in three essential areas in wheeled mobile robots: path planning, simultaneous localization, and mapping and control. It aims to develop an autonomous mobile robot to compete in the micromouse contest [1]. As a basis for the studies, a simulation platform is used to implement and test the developed mapping and path planning algorithms. After the simulation step, a real mobile robot was implemented to validate the methodology in a real maze.

1.1 Motivation

In the last decades robots have become common in industrial and domestic environments performing various tasks. With the emergence of new technologies its use should become even greater. According to the research groups Boston Consulting Group and Tractica [2], the robotic market is expected to grow dramatically in the next few years. Seeking to stimulate research and encourage students in the field of robotics, a wide range of robotic competitions takes place all over the world. Among such competitions the best known is the micromouse.

The micromouse competition began in the late 1970s, being the first competition promoted by the Institute of Electrical and Electronics Engineers (IEEE). In this competition, 8×8 cells mazes were used and the robots achieve speed-run times around 30 seconds. Since then, micromouse competitions have spread all over the world, being especially popular in Japan, Taiwan, Indonesia, the United Kingdom and the United States [3]. Nowadays, 16 $\times$ 16 cell mazes are used and the robots reach speed-run times of less than 5 seconds.

Although it began more than 35 years ago, the importance of the micromouse problem in the field of robotics remains unparalleled, since it requires careful analysis and adequate planning to be solved [4]. In addition to the acquisition of technical skills, group competitions, such as the micromouse, develop teamwork, time management and communication skills.

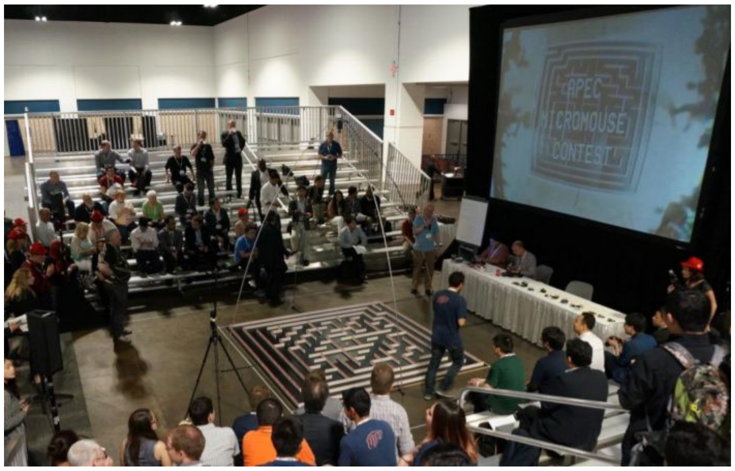


Figure 1.1: APEC 2017 Micromouse Contest Venue [5].

The micromouse is an autonomous mobile robot with reduced dimensions, consisting essentially of one or more drive motors, a steering and turning method, sensors to detect the presence or absence of labyrinth walls, sensors to self-locate, control logic to supervise the actions and keep the vehicle on track, and batteries to provide power [6], capable of exploring, mapping, and racing through a small unknown maze.

In competition, each micromouse begins with some basic knowledge of the labyrinth obtained from the rules [7]. It knows that the labyrinth has a square shape, consisting of 16 $\times$ 16 square cells of 18 cm $\times$ 18 cm, and has walls around the outer perimeter. These walls are 5 cm high, 1.2 cm wide and are painted white to reflect infrared light. The labyrinth floor is painted black to not reflect the infrared light. It also knows that its initial position is one of the four corners of the maze and its goal is to reach the center. Other than that, he knows nothing about the configuration of the walls inside the maze.

Thus, the micromouse's first job is to explore the maze to determine an optimal route from start to finish. As it moves through the unknown labyrinth, it traces its coordinates and maps the walls of each cell into its memory. After a predefined mapping time, it returns to the starting cell. From the initial cell the analysis of the obtained map is carried out and the best route to the center is determined. Then, this route is traveled as fast as possible. The winner of the contest is the micromouse that took the shortest time to arrive from the initial cell to the center.

1.2 Objectives

The main objective of this work is to develop and test a small autonomous mobile robot capable of self-locating, mapping an unknown environment and, based on the map generated, traversing the trajectory between the starting point and the goal in the shortest possible time. To accomplish this main objective, the following sub-objectives are relevant:

- Development of robot control able to move in an unknown environment without colliding.
- Development of a algorithm capable of locate a path to the center of an unknown maze in a limited time.
- Development of strategies for the robot to self-locate and map an unknown maze.
- Develop a path-planning algorithm that generates the shortest path between the starting point and the goal based on the map obtained.
- Development of a robot control capable of performing the trajectory generated in the shortest possible time.
- Implement the methodology developed in SimTwo simulation environment.

- Assemble a robot capable of accomplishing the required tasks of the micromouse competition.

- Implement the methodology developed in a real labyrinth using the assembled robot.

1.3 Dissertation Outline

This document is divided into 7 chapters, which present the micromouse competition, the concepts on which this work is based, the assembled robot, the simulation environment, the methodologies developed, and the results obtained from its application. The following is a brief description of the contents of the chapters that make up this dissertation.

The introduction is described in Chapter 1, which presents the proposal of the work along with the description and rules of the Micromouse Portuguese contest that guide this project.

Chapter 2 presents a few robotic competitions demonstrating the areas of study benefited by being part of the requirements necessary in such contests. Following, real applications of mobile robots are presented, showing the applicability of mobile robots in various tasks. Then, a theoretical revision of the concepts on which this work is based, such as simultaneous localization and mapping, wheeled mobile robots geometry and path planning, is carried out.

Chapter 3 presents the robot assembled to solve the micromouse challenge demonstrating the kinetic model and structural and electronic composition of the developed micromouse robot.

Chapter 4 describes the simulation environment emphasizing the generation of mazes, the simulation model of the robot, its control and, the presentation of the results in real time.

Chapter 5 describes the algorithms implemented for the robot to be able to perform the proposed objectives. These algorithms control the robot, map the surroundings, localize the robot in the maze and plan the best path between the starting point and the goal.

Chapter 6 contains the results of the different methodologies of locomotion and path planning in different mazes. The locomotion algorithms are evaluated by analyzing the percentage of the maze mapped in a period of six minutes and whether the center was reached. Meanwhile, the path planning algorithms are evaluated for the processing time and the length of the generated path. Finally, the times obtained in the evaluation runs are presented.

Lastly, Chapter 7 presents the conclusions obtained by analyzing the set of developed methodologies and future works.

Chapter 2

Related Work

Taking into consideration the increasing use of robots in the most varied applications, this chapter seeks to elaborate, based on the current scientific scenario, the concepts and applications required in the development of this project. It will be analyzed the related work in the area of wheeled mobile robots such as, robotics competitions, real applications, geometries, localization and path planning.

2.1 Robotics Competitions

Robot competitions bring together researchers, students, and enthusiasts in the pursuit of a proposed technological challenge [8]. Such competitions are an excellent way to foment research and to attract students to technological areas [9] introducing new technologies, teamwork [10] and even developing solutions to real challenges in industry.

Robotics competitions present problems that can be used as reference to evaluate and compare the performances of different approaches [11]. This possibility of performance comparison leads to advances in several areas of robotics. It is worth highlighting the improvements in the autonomous control of Unmanned aerial vehicle (UAV) obtained from the competition Association for Unmanned Vehicle Systems International (AUVSI) and in autonomous vehicles generated from the Association for the Advancement of Artificial Intelligence (AAAI) mobile robot competition [8].

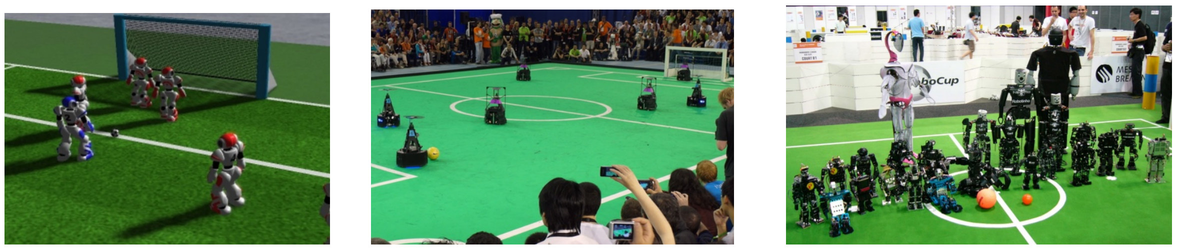
This section presents some of the best known competitions along with the main areas of study required to meet the challenges proposed.

2.1.1 RoboCup Soccer Competition

RoboCup is an international joint project to promote Artificial Inteligence (AI), robotics, and related fields. The soccer game has been selected for the competition because of the requirement of multiple players to cooperate in a dynamic environment, this way, the competition does not evaluate isolated individuals, but robots as a team. The soccer competitions at RoboCup are held in five leagues: humanoid, four-legged, middle size, small size, and simulation.

Different research issues are addressed to the different leagues. In the simulation league (Figure 2.1a), advanced team play, and learning are required. The small and middle size leagues are played by robots with wheels, Figure 2.1b. In such leagues the robot construction, the perception of the field situation, and the implementation of basic soccer skills are the center of the activities. The four-legged league is played by Sony Aibo dogs and the focus of research occours in perception and behavior control, since the robot presents 18 degrees of freedom [8].

The Humanoid League is the most challenging RoboCup soccer league (Figure 2.1c). It requires the robots to have a human-like body, be fully autonomous, and the only allowed modes of locomotion are bipedal walking and running [12].



(a) (b) (c)

Figure 2.1: RoboCup soccer competitions : (a) Simulation league. (b) Middle size league. (c) Humanoid league.

2.1.2 Urban Search and Rescue Competitions

The recent advances in robotics platforms and intelligent software combined with the occurrence of disasters, such as earthquakes and hurricanes, boosted the research of the Urban Search and Rescue (USAR) robotics. USAR is a branch of robotic rescue that concentrates on victim detection and removal from man made structures, such as collapsed buildings [13].

In order to stimulate research in the field, competitions such as [14], [15] were proposed. In these competitions teams of intelligent robots are placed in confined spaces, classified according to their physical complexity, in order to locate and rescue possible persons. Figure 2.2 shows an example of USAR robot. To obtain the desired result, the robot must have autonomy of navigation, be able to search the environment, work cooperatively and deal with the uncertainty of the sensing, since the environment is unstable and only partially known.

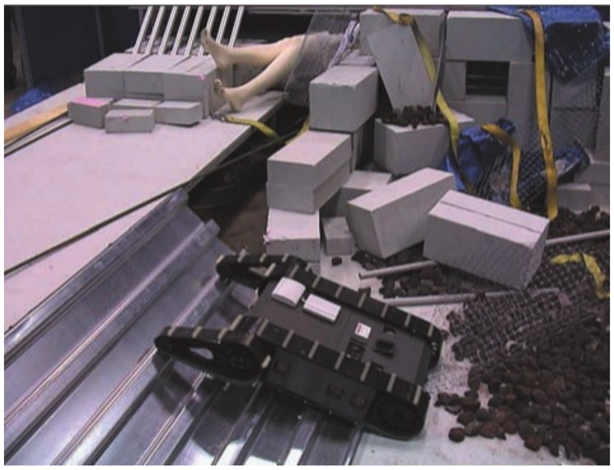


Figure 2.2: A USAR robot operating in the NIST arena [13].

2.1.3 Robot@Factory

The Robot@Factory competition attempts to recreate the problems that one autonomous robot will face during its use in a factory. Therefore, competition occurs in an emulated factory environment. In this case, the factory is comprised of a supply warehouse, a final product warehouse and eight processing machines, as shown in Figure 2.3.

The robot's task consists in carrying the material from the supply warehouse to the machines and then to the final product warehouse. To do so, the autonomous robots must be able to collect, carry and put materials in the right position, locate and navigate in the given environment, as well as avoid collisions with walls, obstacles and other robots [16].

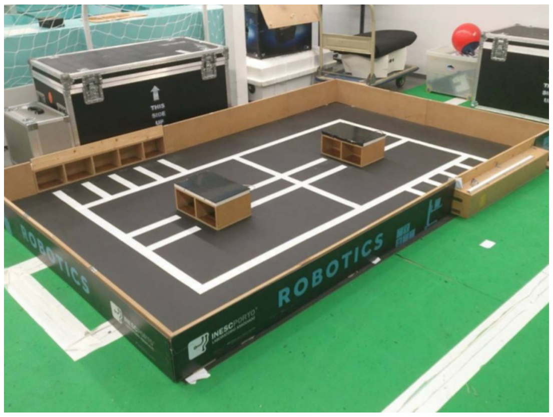


Figure 2.3: Robot@Factory competition arena [16].

Based on this competition, Robot@factory Lite starts in 2019. This competition has the same proposal as the Robot@factory, but without the mechanical problem of controlling the fork.

2.1.4 DARPA Grand Challenge

The DARPA Grand Challenge was launched in 2003 to spur innovation in unmanned ground vehicle navigation. The goal of this Challenge was to develop an autonomous ground vehicles capable of traversing a desert course up to 175 miles long in less than 10 hours [17].

The main technological challenge in the competition was to build a highly reliable system, capable of driving at relatively high speeds through diverse and unstructured off-road environments, as can be seen in Figure 2.4. Since the route information is displayed some time before the race, no global path planning is required. This competition has brought several advances in the field of autonomous navigation, especially in high-speed location testing, real-time crash prevention and stable vehicle control on slippery and steep terrain.

Figure 2.4: DARPA Gran Challenge. Adapted from [17].

2.2 Mobile Robots Applications

According to the dictionary, robot is a machine designed to execute one or more tasks automatically with speed and precision. The first robots date back to the 1970s and consisted of stationary, non-programmable, sensorless electromechanical devices. With the advance of technology, new features were added to these robots, such as programmable controllers and sensors, to allow the robot to perceive its environment[18] and decide the action to be performed. Nowadays, robots can be stationary or mobile, autonomous, have sophisticated programming, speech recognition, and other advanced features.

In general, robots were developed to perform tasks that the human being does not perform, for reasons of unhealthiness, incapacity or disinterest [19]. Robotic systems are usually implemented to execute tasks with higher quality, speed and at a lower cost compared to a human [20]. Mobile robot with wheels is the focus of this work. A mobile robot with wheels is applied to problems that require the movement of the robot inside an environment (Figure 2.5) , which can be simple or complex, known or unknown, static or dynamic.

The world market of mobile robotics is expected to increase substantially in the next

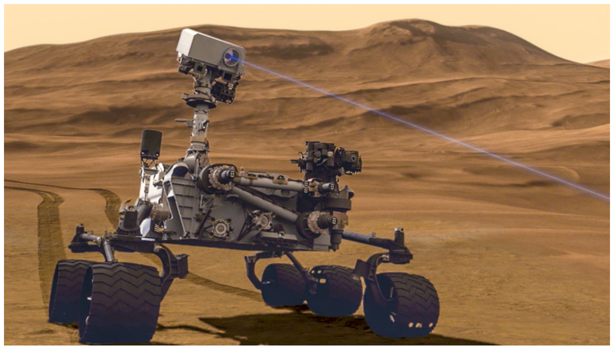


Figure 2.5: Curiosity - Robot for exploration of the planet Mars (NASA). Adapted from [21].

20 years. the forecasts of all the major robotics research institutions clearly indicate that the world market of service robotics, especially ground mobile robots, is expected to increase dramatically over the next years, surpassing the market of industrial robotics in terms of units and sales [22]. Among the fields of application are homeland security [23], surveillance [24], [25], demining [26], agriculture [27], and others.

2.3 Locomotion of Wheeled Mobile Robot

Several mechanical robot architectures have been proposed by academic and industrial researchers, each presenting advantages and drawbacks according to its application [22]. Consequently, when designing a new mobile robot for a specific application, it is necessary to evaluate which technological solution best fits the required conditions. Some of the key conditions to be evaluated are the operating environment and requirements for the robot to perform its task optimally.

Considering such conditions in the context of this work, where the environment is flat and the robot must operate at high speed, wheeled mobile robots prove to be the best option, since they can reach high speeds with low power consumption and can be guided by controlling a few active degrees of freedom [28]. In this section the main types of locomotion developed for mobile robots with wheels are presented.

2.3.1 Ackerman Steering Geometry

The Ackerman steering geometry is characterized by all wheels having their axis arranged as a radius of a circle, with a common center point. As the rear wheels are fixed, this point must be defined in an extended line passing through the rear axle. In order to intercept the front wheels tangentially to the axis of their movement, it is necessary that such wheels have different angles, the inner front wheel must have an angle of rotation greater than the outer front wheel. Such a difference occurs using rods and two extra pivots forming a trapezium. This configurations is presented in Figure 2.6.

The advantage of this system is to eliminate the need for side-slip or skidding, reducing the effort of the engines when making the turns. However, this system requires a minimum radius of curvature that makes maneuvering difficult, especially in tight environments.

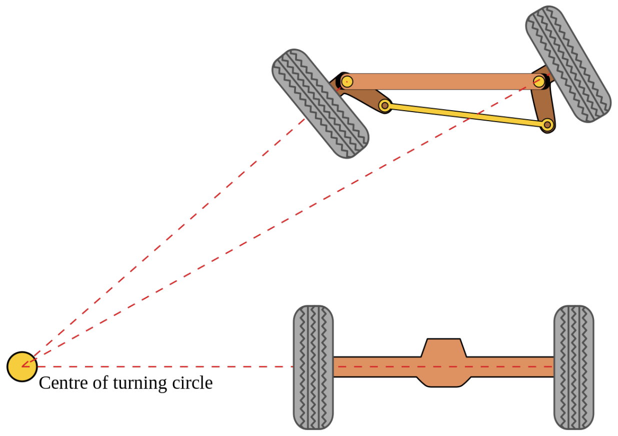


Figure 2.6: Ackerman steering geometry realizing a turn.

2.3.2 Omnidirectional Geometry

Robots with omnidirectional geometry usually have three or four wheels, each controlled by a motor. This geometry is characterized by its movement having no non-holonomic constraints, in other words, the robot is able to move in all directions. To perform such movements, the wheels require a special shape that allows movement on two different axes. The change between the axes of movimentation occurs when different speeds and directions are set for these wheels. Figure 2.7 shows an example of robot with omnidirectional geometry.

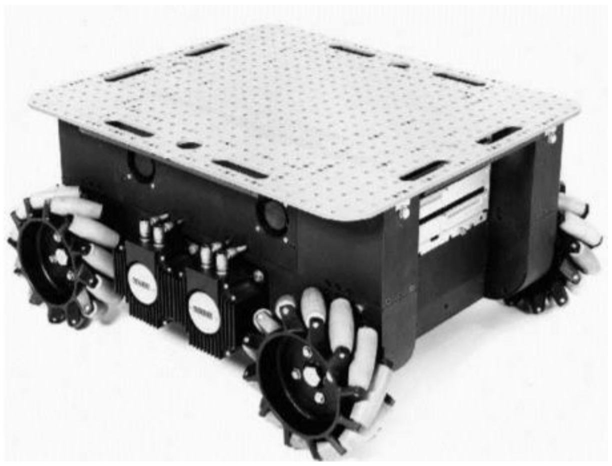


Figure 2.7: Robot with omnidirectional geometry (Uranus) [20].

2.3.3 Differential Geometry

Robots with differential geometry are composed by two main wheels and one or more castor wheels, that is, wheels that can freely move in two different axis, to support the robot and prevent it from unbalancing and possibly falling. The main wheels are positioned on the same axle and are controlled by independent motors. The speed variation of the wheels controls the direction of the robot, this allow the robot to turn without change it axis central position. However, unlike the omnidirectional robot, this configuration presents non-holonomic constraints. Figure 2.8 presents Cyo, an example of robot with differential geometry that can vacuum and make deliveries in the home.

2.4 World Representation

For an autonomous robot mobile be able to perform the assigned tasks, it must be capable of learn and maintain models (maps) of the environment [29]. Without this maps the calculation of the relative position of the objects around the robot and the determination of the routes become more difficult and imprecise, causing imperfect executions and possible

Figure 2.8: Robot with differential geometry (Cyo, produced by Probotics, Inc).

collisions [30]. To faithfully represent the location where the robot operates two major paradigms for mapping indoor environments was produced: Grid-based and topological paradigms.

2.4.1 Topological

Topological approaches represent the robot environment per graph, as proposed by [31] and [32]. This chart is made up of nodes and lines. Where nodes are positioned at identifiable landmarks and locations, such as intersections, and if there is a direct path between nodes, a line connects them. Figure 2.9 presents a simple topological map representation.

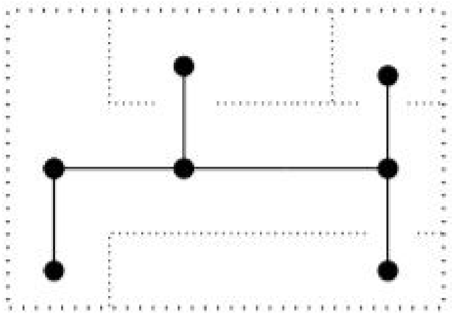


Figure 2.9: Topological map representation [33].

The main advantage of using the topological representation is the size of the generated map, since the resolution of the map corresponds directly to the complexity of the [29] environment. This compact representation allows for quick planning and provides a more natural interface for human instructions such as "Go to Room A". Another advantage lies in the fact that topological approaches do not usually require the exact determination of the geometric position of the robot, being able to recover from odometry errors.

However, accurate and consistent topological maps are difficult to learn in large-scale environments, particularly if momentary sensor data is highly ambiguous. In such case the topological approach often have difficulty determining if these places are the same or not.

2.4.2 Grid-based

Grid-based approaches represent environments by evenly-spaced grids. Each cell represents a fixed portion of the environment and stores a value that indicates the state of such cell [29]. This state indicates whether or not the center of the robot can be moved to the center of that cell. Figure 2.10 presents a simple Grid-based map representation.

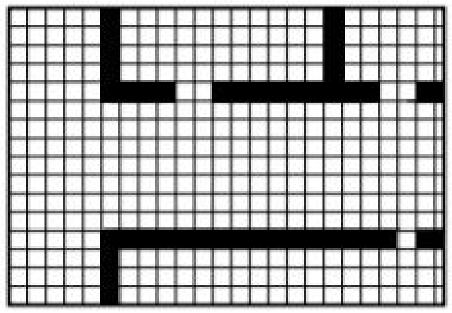


Figure 2.10: Grid-based map representation [33].

Such grid mapping approaches are easy to build and maintain (if the environment changes, it is only necessary to change the value of some cells) in large-scale environments. It also avoids the problem of ambiguity of the sensors that the topology approach presents, since the places location in the map are based on the robot's geometric position within a global coordinate frame.

On the other hand, grid-based approaches suffer with their enormous complexity of space and time. Once the size of the grids must be small enough to capture all the important details of the world, drastically increasing the number of cells according to the complexity of the environment. Another problem is the need to accurately determine the position of the robot, so the odometer and sensors must be calibrated to avoid the accumulation of errors.

2.5 Simultaneous Localization and Mapping

The problem of Simultaneous Localization and Mapping (SLAM) has attracted immense attention in the mobile robotics literature. This problem asks if it is possible for a mobile robot to be placed at an unknown location in an unknown environment where the robot will, incrementally, build a consistent map of this environment while simultaneously determining its location within this map [34]. Such capacity is considered to be a key prerequisite of truly autonomous robots [35].

The solution to the SLAM problem is of high importance in a range of applications where absolute position or precise map information is unobtainable, including, among others, autonomous planetary exploration, sub-sea autonomous vehicles, autonomous airborne vehicles, and autonomous all-terrain vehicles [36].

In general, three philosophy are used to solve this problem. The most popular of these is the estimation-theoretic or Kalman filter based approach. The popularity of this approach is due to it directly provides both a recursive solution to the navigation problem and a means of computing consistent estimates for the uncertainty in vehicle and map landmark locations on the basis of statistical models for vehicle motion and relative landmark observations [36]. An example of applying the Kalman filter can be found in [19].

The qualitative philosophy seeks to avoid the need of estimates of absolute position and precise measures of uncertainty. Instead, it employs qualitative knowledge of the relative location of landmarks and the vehicle. Several approaches have been developed based on this philosophy, as presented in [37] [38]. In these approaches, several advantages were observed in comparison to the theoretical-evaluative methodology, especially in terms of limiting the need for precise models and computational requirements.

The third philosophy uses an essentially numerical or computational approach to solve the problem, however its not rigorous as the Kalman filter. Such approaches are based on the use of landmark matching, global map registration, bounded regions and other measures to describe uncertainty, as presented in [39] with a grid based approach.

2.6 Path Planning

Since the 1980s, mobile robot motion planning problems have become an important research topic that has attracted the attention of many researches who have worked extensively to obtain efficient methods to solve these problems [40]. The choice of the path planning algorithm is one of the most important assignment on the robot navigation, since it will determine the efficiency of the robot in performing the given task.

The mobile robot path planning task is to find a collision-free route, through an environment with obstacles, from a specified start location to a desired goal destination while satisfying a certain optimization criteria [41]. The path planning methods can be classified according to the type of environment where the robot is located, static or dynamic, and the quantity of information that the robot have of the surroundings, which can be known or unknown.

Below, will be presented some of the main methodologies used to generate a collision-free path between two defined locations. Since this work contemplates the maze environment, where the obstacles are only static, only the path planning in static environments are presented.

2.6.1 Cell Decomposition

The methods of cellular decomposition are the most studied methods in mobile robotics. These methods are based on the discretization of the environment, the entire analyzed space is divided into non-overlapping regions (cells) whose union results exactly in the previously discretized environment. The result is a graph in which each cell is adjacent to another cell. The methods to cross from one cell to another are called connectivity graph. Based on such graph a path is generated, this path consists in the sequence of cells the robot should transverse to reach the goal.

Below, three of the most known approaches to environmental discretization are presented along with the advantages and disadvantages of their use.

2.6.1.1 Approximate Cell Decomposition

Proposed by [42], approximate cell decomposition is elaborate by laying a grid, formed of cells with predefined shape and size, over the entire environment. The map is then assembled by marking the cells that contain the obstacles and the center of each cell becomes a node in the search graph that will search for the best paths.

This method is called "approximate" because the boundaries of physical objects in the world do not necessarily coincide with the predefined cell boundaries [33]. In this approach the resultant route is usually a conservative estimate, since the entire cell containing a small object is labeled as occupied despite having free space. Such conservative estimate might present errors once small passages end up labeled as obstacles, as shown in Figure 2.11.

The advantages of this approach lie in the easier implementation compared to other algorithms, it is simple to apply to the environment and it is flexible. Cell size can be adjusted to control computational time and map quality. If the cell size is incremented, computational time and map details are decreased, and if size decreases, computational time and map details increase.

2.6.1.2 Adaptive Cell Decomposition

Adaptive cell decomposition is utilized to save processing time and memory storage by reducing the number of cells in open space. This approach relies on the fact that much of the information in open spaces is redundant in a regular cell decomposition [33].

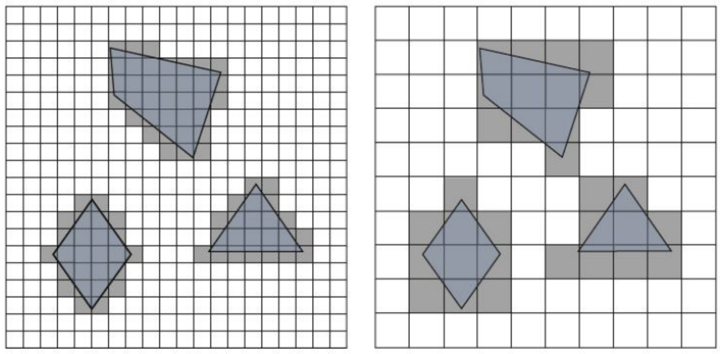


Figure 2.11: Cell decomposition with different resolutions [43].

One of the possible adaptive cell decomposition is the quadtree decomposition, as presented in [44]. This method begins by imposing a large size cell over the entire planning space. If a grid cell is occupied, it is subdivided into four equal parts. This process repeat until each of the cells is either entirely empty or entirely full. The result is a map with various grid cells size, as shown in Figure 2.12, and well defined obstacle boundaries.

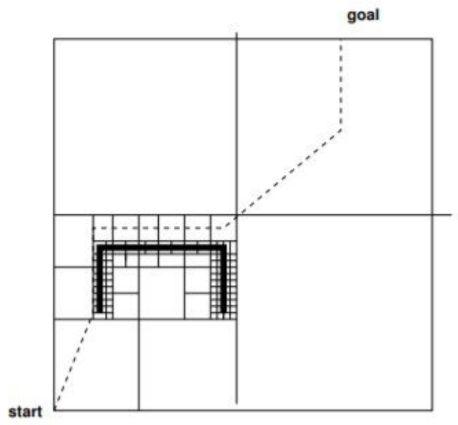


Figure 2.12: Quadtree representation [33].

The problem with such approach is the difficulty in providing near optimal paths, often result in jagged paths, and in high clutter environments it can be less efficient than regular grids.

2.6.1.3 Exact Cell Decomposition

Exact cell decomposition is a solution to some problems that occur in regular grids. In this approach, the cells do not have predefined size or shape, they are based on the exact information of the obstacles. The result is a map almost identical to the real environment, which will always find a way, if it exists.

However, this approach is quite difficult to implement since there is no simple rule on how to decompose space into cells. Another problem is the sub-optimal paths generated, this occurs due to the details resides on the obstacles, not the free space. Examples of exact cell decomposition are found in [45].

2.6.2 Roadmap Methods

The Roadmap methods are the second most studied methods in mobile robotics. Roadmap are graphs that represent how to go from an initial position to the goal. This graph is based on nodes and edges, where the nodes represent certain positions and the edges connect these nodes creating a path. The best path is a combination of edges that connect the starting point to the goal. There are three main approaches of this methodology: visibility graph, Voronoi diagrams and Probabilistic Roadmaps.

2.6.2.1 Visibility Graphs

The main idea of the visibility graph method is that if there is a collision-free path between two points, then there is a polygonal path that bends only at the obstacles vertices [46]. Basically, this method consist of straight lines connecting nodes, that are positioned at the start point, goal and the vertices of all obstacles, without crossing the interior of them. These straight lines make up the paths on which the robot may transverse [33]. Figure 2.13 shows a complete visibility graph

The main disadvantage of such method is that the calculated paths are tangential to the obstacles and the robot will brush or even collide against them. However, such problem can be fixed by enlarging the obstacles, creating a safety margin, although this

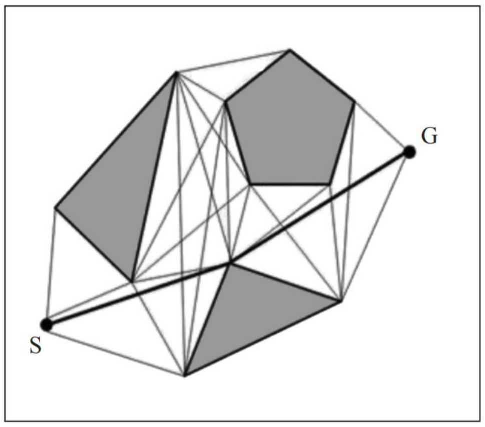


Figure 2.13: Complete visibility graph [47].

results in incompleteness and inefficiency of the planner.

2.6.2.2 Voronoi Diagrams

The Voronoi diagram is the set of points where the distance to the two closest objects, in this case obstacles, is the same [48]. The set of points where this edges meet are called vertices and consist on the possible roads to the robot reach the objective. Figure 2.14 present an example of Voronoi diagram application.

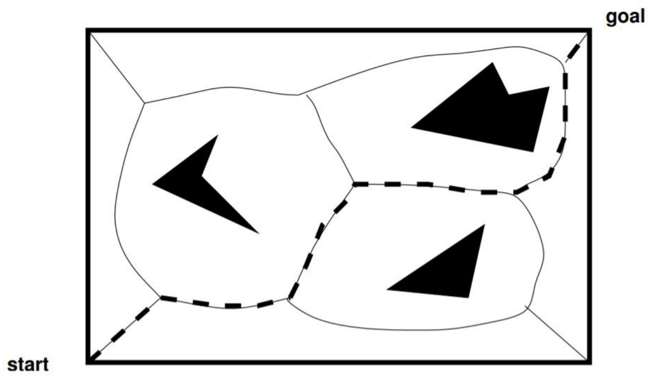


Figure 2.14: Voronoi diagram [33]

Voronoi paths are by definition as far from obstacles as possible. This makes the Voronoi diagram method safe for the robot to move around without possible collision, but the paths generated are inefficient.

2.6.2.3 Probabilistic Roadmaps

The global idea of Probabilistic Roadmap is to pick a collection of random configurations in the free space [49]. In other words, the Probabilistic Roadmap create a discrete version of the free space by randomly sampling it. Such discretization reduces the search space and consequently accelerate the planning process.

The path planning process can be divided in two phases: construction of the roadmap and path query. To construct the map random points in the space are chosen and added to a list, if one of these points is inside an obstacle, it will be discarded. With these points the mapping algorithm attempts to connect them, usually simply by using straight lines between the points and not connecting them if an obstacle is in the way. In the query phase, when the robot need a path between two points, the algorithm uses the connections created in the fist phase to find the nodes that leads into the lowest cost path [33]. Figure 2.15 show an example of probabilistic roadmap generation.

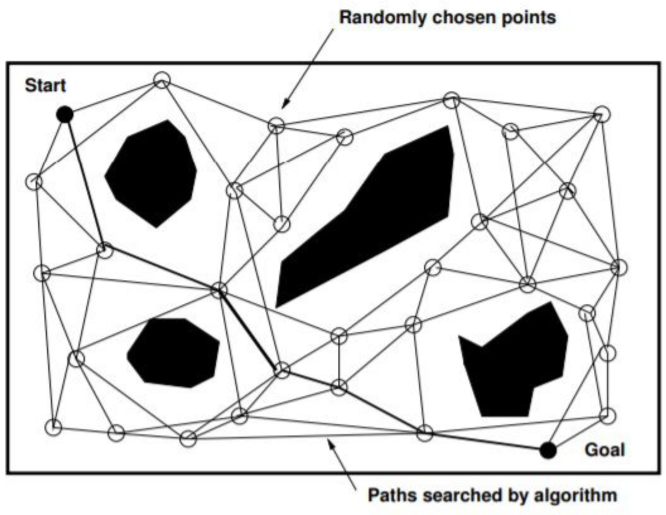


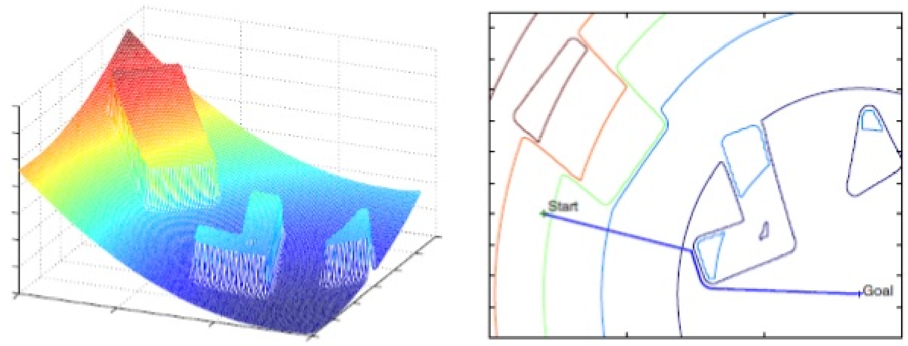
Figure 2.15: Probabilistic roadmap [33]

The main problem with the Probabilistic Roadmap is inefficiency in narrow spaces. Once the points are chosen randomly, the chance of a random point being placed in a tight space is low, and if there is no point, no connectivity will be established in this space. Such problem can be fixed by increasing the number of points on the space, but this also increases the processing time and make the planning algorithm more complex.

2.6.3 Potential Fields

Initially proposed by [50] and further developed in [51] the potential fields methods are quite different for the previously presented. This method instead of trying to map the environment applies a mathematical function over the entire space. Such function acts like an artificial potential field in which the target is an attractive pole and the obstacles are repulsive surfaces [40].

In such method the robot is treated as a point under the influence of the generated fields and its behavior is similar as an electron in an electric field [33]. At every spot in the environment the resultant force of the fields on the robot define the direction of the robot's motion.



(a) Environment with three obstacles. (b) Resulting path.

Figure 2.16: Potential field algorithm [52].

The potential field methods are relatively easy to implement and are computationally efficient. However, the major problem is that robots are often trapped into a local minima before reaching the destination. To improve its efficiency this method is combined with many other computational methods. In [53] and [54] the potential field method was integrated with the simulated annealing algorithm to escape the local minima problem.

2.7 Trajectory Planning

Trajectories planning are crucial in robotics, because defining the times of passage at the via-points influences not only the kinematics properties of the motion, but also the dynamic ones. The trajectory planning generate the reference inputs for the control system of the robot, so as to ensure that the desired motion is performed. Usually, the algorithm employed for trajectory planning takes as inputs the path generated by the path planner, as well as the kinematic and dynamic constraints of the robot. The output of the trajectory planning is given by the trajectory of the joints, or the robot, in form of a sequence of values of position, velocity and acceleration [55].The optimization criteria of the trajectory planning define how the robot will interact with the environment while the selected path is traveled.

The trajectory planning methodologies can be classified according to the set optimization criteria. The most common are minimum execution time, energy and jerk. In addition, some hybrid optimization have been proposed, such as time-energy optimal trajectory planning.

This section presents three well-known route planning algorithms. These algorithms return the set of cells to be visited to reach the goal. Based on the location of these cells and the position of the robot, the trajectory to be followed is defined.

2.7.1 A* Algorithm

A-Star algorithm is one of the best-known path planning algorithms, which can be applied on metric or topological configuration space [56]. It is based on a cell map where each cell (position) represents a node [57]. A* explores the environment by calculating a cost function for each possible next position to be searched, this function $f(n)$ is used to determine the order in which the nodes will be searched. Then the lowest cost position is added to the search space. Adding this new location to the search space generates more path possibilities [58].

The $f(n)$ cost is a sum of two functions, $f(n) = g(n) + h(n)$, where $g(n)$ is the length of the path from the origin to a node n through a specific cell path and $h(n)$ is the heuristic distance of the cell to the target node. In this project only horizontal and vertical movements were used. The Figure 2.17 shows an application of the A* algorithm in its first search, showing $h(n)$ in purple, $g(n)$ in blue and $f(n)$ in black.

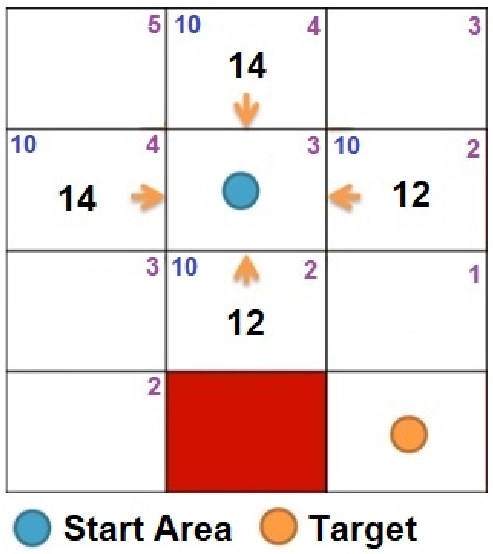


Figure 2.17: A Star Algorithm

The memory requirements for the algorithm is composed by two lists. The open list (O_{list}) contains the nodes that are candidates for exploration and the closed list (C_{list}) contains the already explored nodes. The parent node information is stored along with each node that is stored in (O_{list}) and (C_{list}). To obtain the best path, the lowest cost node of the (O_{list}) is selected. The neighbors of this node are processed and inserted into the (O_{list}) if they are not there, and the initial node is moved to the (C_{list}). If any of the neighbor nodes are already contained in (O_{list}), the cost of the node already inserted in the list is compared to the newly obtained cost. If the new cost is smaller, the parent node will be changed to the newly obtained node. This procedure is repeated until the target node is reached or the (O_{list}) becomes empty which means there is no solution. In the end, the values of the parents of each node, from the objective to the beginning node, are consulted. This list of nodes give the shortest path.

2.7.2 Dijkstra's Algorithm and Best-first Search Algorithm

Predecessors to A-Star, these algorithms work identical to A * with the variation of using only a partial of its search function $f(n)$. Dijkstra is an uninformed algorithm that searches the graph based only on the cost of each move $f(n) = g(n)$. This way, the search usually demand a longer time, but always return in the best way.

Meanwhile, the best-first search is an informed algorithm that performs its search by checking the cells with the greatest promise of being close to the target [59]. The search function is based only on the heuristic distance $f(n) = h(n)$. This result in a fastest search, however it may return suboptimal paths.

Chapter 3

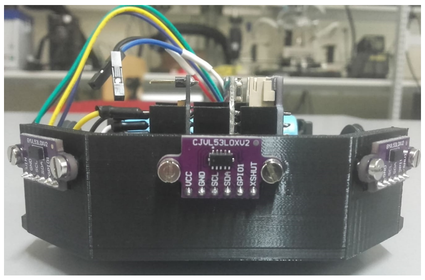
Real Robot Description

The mobile robot shown in the Figure 3.1 was designed and developed for the purpose of completing a micromouse challenge. Its structure is designed with dimensions that meet the rules of such competition, the robot must be no more than 250 mm wide and 250 mm long [7], and with components that allow the robot to locate, move and identify the environment around it, necessary conditions to map the labyrinth, to plan the best way between the starting point and the goal and to go through this route.

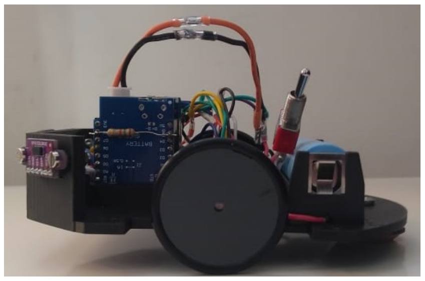
This section describes the settings and features that compose the real robot. Initially, it will be approached the structural part of the robot, highlighting the development of the central plate of the robot and the reasons for the differential geometry to be chosen for the development of the micromouse. Then, the cinematic model of the differential robot is described. Subsequently, the electronic scheme of the robot is presented along with a description of the electronic components used. Finally, the software used in the robot development is described.

3.1 Structural Constitution of the Robot

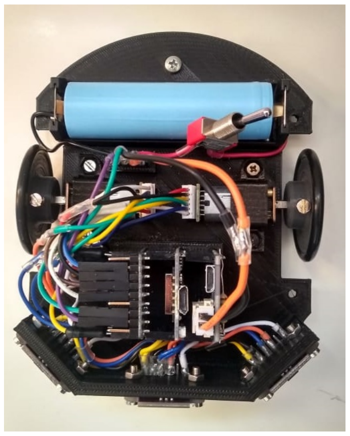
The micromouse robot is designed to traverse a narrow maze at high speed. In order to achieve such performance, two factors have high importance: the robot weight must not be too high and its center of mass must lie within the triangle formed by its three wheels.



(a)



(b)



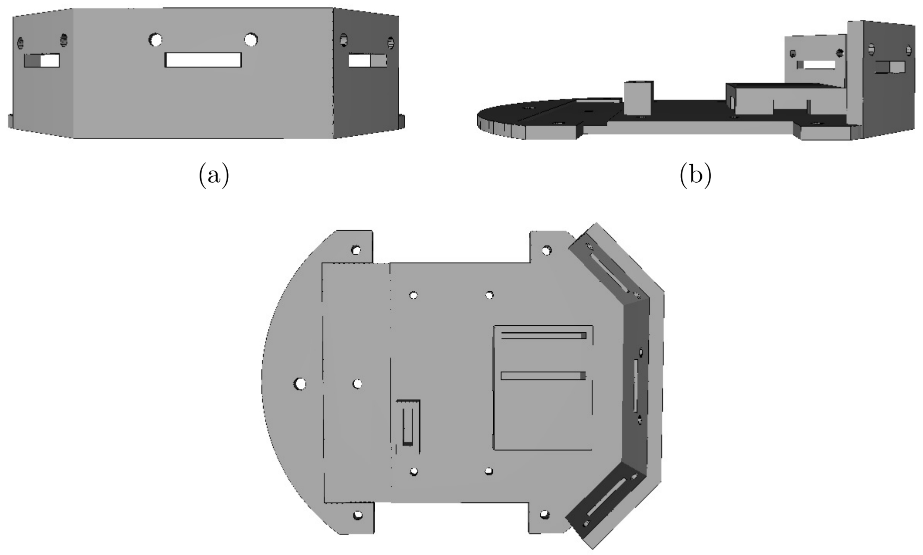
(c)

Figure 3.1: Assembled micromouse robot : (a) Frontal view. (b) Side view. (c) Top view.

To obtain such characteristics, a base plate has been specially developed, as shown in Figure 3.2.

Seeking to obtain greater stability and to increase the reliability of the collected data, grooves to fit the components were inserted in the plate. Such grooves were designed according to the dimensions of the components used. The plate also has a bumper where the distance sensors are placed. This plate was designed using Open SCAD and printed on the 3D printer with 50 percent density so as to be lightweight and sturdy.

For the robot architecture, the differential geometry was selected. This geometry presents several advantages compared to the architectures presented in the section 2.3, highlighting: the ability to rotate without changing the position of its central axis, a requirement to maneuver in the narrow maze environments, mechanical and control simplicity, low maintenance rates and high odometric accuracy, especially in comparison with



(a) (b) (c)

Figure 3.2: Base plate designed for the micromouse robot : (a) Frontal view. (b) Side view. (c) Top view.

the omnidirectional architecture.

Following the concept of differential architecture the robot has three wheels. Two of them are connected to DC motors, while the other is castor wheel which has the support function. The Table 3.1 presents the dimensions of the mobile robot structure.

Table 3.1: Dimensions of the micromouse robot

Robot Description	Dimension	Unit
Width	0.096	m
Length	0.120	m
Wheel diameter	0.032	m
Wheel thickness	0.008	m
Robot mass	1.25	kg

3.2 Kinematic Model of Differential Robot

Kinematics is the study of how mechanical systems behave. In mobile robotics, it is necessary to understand the mechanical behavior of the robot to properly design it to perform the assigned tasks and to understand how to develop its control software [20]. In this context, it is important to know robot's workspace, as it defines the range of possible poses that the mobile robot can reach, and its controllability, which defines possible paths and trajectories in its workspace.

This section presents the mathematical modeling that represents the kinematics involving the robot.

3.2.1 Pose

Throughout the analysis, the robot was modeled as a rigid body on wheels, operating on a horizontal plane (2D plane). The total degrees of freedom of this robot on the plane is three, two for position in the plane and one for orientation along the vertical axis, which is orthogonal to the plane.

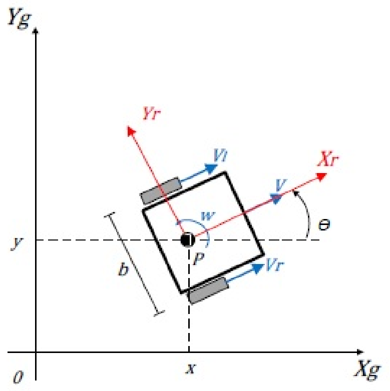


Figure 3.3: Robot in the plane representing the global robot reference frame. Extracted from [19].

To specify the position of the robot in the plane, a relation was established between the global reference of the plane and the local reference of the robot. Figure 3.3 demonstrates these references, where Y_g and X_g define the arbitrary inertial basis in the plane. The position of the robot in the global reference frame is specified by coordinates x and y, and the angular difference between the global and local reference frames by θ. The robot pose (ε_g) is described as

$$\varepsilon_g = \begin{bmatrix} x & y & \theta \end{bmatrix}. \tag{3.1}$$

The relation between the global and the robot frame is defined by the rotation matrix

$$R(\theta(t)) = \begin{bmatrix} cos(\theta(t)) & sin(\theta(t)) & 0 \\ -sin(\theta(t)) & cos(\theta(t)) & 0 \\ 0 & 0 & 1 \end{bmatrix}. \tag{3.2}$$

This matrix can be used to map motion in the global reference frame (X_g, Y_g) to motion in terms of the local reference frame (X_r, Y_r).

3.2.2 Linear and Angular Velocity

The velocity of each wheel is controlled by a DC motor. Deriving the state of the robot in the global reference frame, the speed relation is

$$\dot{\varepsilon} = \begin{bmatrix} \dot{x(t)} & \dot{y(t)} & \dot{\theta(t)} \end{bmatrix} = \begin{bmatrix} cos(\theta(t)) & 0 \\ sin(\theta(t)) & 0 \\ 0 & 1 \end{bmatrix} \cdot \begin{bmatrix} V(t) \\ W(t) \end{bmatrix} \tag{3.3}$$

where the input variables of this system $V(t)$ and $W(t)$ are, respectively, the linear and the angular velocities of the robot, obtained by

$$V(t) = \frac{V_r(t) + V_l(t)}{2} \tag{3.4}$$

and

$$W(t) = \frac{V_r(t) + V_l(t)}{b} \tag{3.5}$$

where V_r and V_l are, respectively, the speed on the right and left wheel, and b is the distance between the traction wheels of the robot at its point of contact with the ground.

3.3 Electronic Hardware

The mobile robot, where the control, mapping, and path planning methods were implemented, has an elaborate electronic structure to move through the maze autonomously. To this end, the developed micromouse incorporated in its structure some electronic components, batteries, a Wemos D1 mini [60], Wemos shields [61], [62], sensors and two DC motors. The block diagram of Figure 3.4 shows the components integrated in the mobile robot. The electronic architecture of the robot is designed to be compact in order to fit the small base, presented in the Section 3.1, and to have low battery consumption.

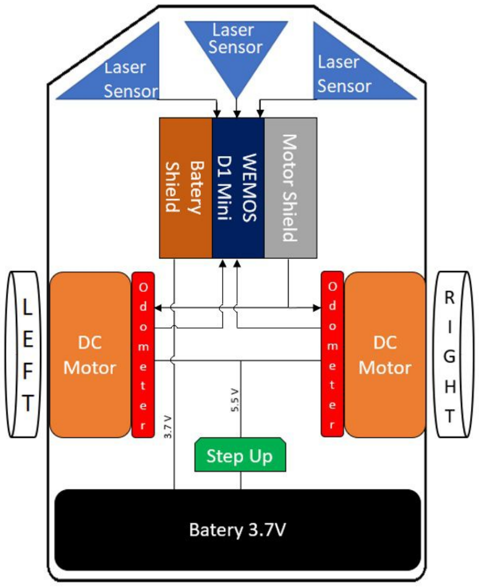


Figure 3.4: Micromouse electronic architecture.

This section presents a brief description of the electronic components used in the development of the robot.

3.3.1 WEMOS D1 Mini

The main electronic hardware component on the mobile robot is the WEMOS D1 Mini (Figure 3.5). This is a miniature wireless microcontroller development board based on the popular ESP8266 microcontroller. The board has a Tensilica L106 32-bit RISC processor running at 80 MHz, 16 MB of Flash memory, 11 digital input/output pins, 1 analogue input pin and an external antenna connector.

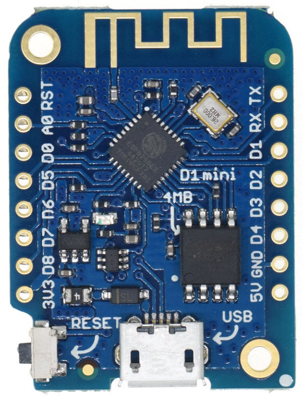


Figure 3.5: WEMOS D1 Mini.

This board is responsible for the entire control system, gathering information from the lasers sensors and odometers to provide the robots location, map and plan the paths. The microcontroller is powered by a 3.7 V battery through a battery shield.

3.3.2 Motor Shield

To control the DC motors a WEMOS I2C Dual Motor Driver Module was used (Figure 3.6). This module is able to drive up two DC motors from the WEMOS D1 Mini. Using Pulse Width Modulation (PWM), the motor shield can control independently the speed and direction of each motor connected. PWM is a simple method of controlling analog devices through a digital signal by changing or modulating the pulse width [63]. The speed control of the motors occurs by changing the average voltage supplied to them, this voltage is adjusted by changes in the duty cycle.

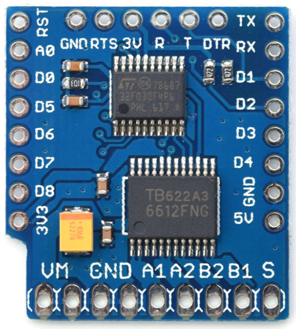


Figure 3.6: Motor shield.

The shield provides the side grooves that allow it to be easily inserted and removed from the WeMos Mini. Its control is performed through the I2C interface of the WeMos D1 mini, by default the address 0x30 is used in the communication.

3.3.3 Battery Shield

The WEMOS D1 mini lithium battery shield, shown in Figure 3.7, is used to power the WEMOS D1 mini from a lithium battery up to 4.2 V. This shield contains a DC-DC converter to step-up the voltage supplied by the battery to 5 V, providing up to 1 A to the WeMos mini and its shields.

This shield also allows recharging the battery, simply connect the shield to a USB power source using a mini USB cable. Two LEDs on the shield indicate battery charge status.

3.3.4 Step-up and motor

The robot uses two DC micro metal gear motors, each coupled to one of the traction wheels. These motors have a 50:1 gearbox in order to smooth out the robot's motion

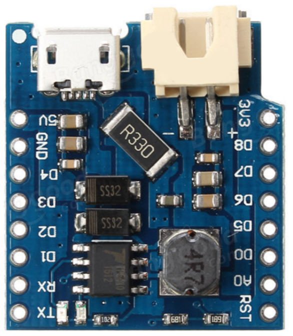


Figure 3.7: Battery shield.

and increase engine torque, allowing a larger weight limit for the robot. These motors also comes with an integrated encoder that measures the motor speed in real time. The average resolution of this encoder is 58.94 pulses per revolution. The motors specifications are presented in the Table 3.2.

Table 3.2: Micro metal gear motor specifications

Specification	Micro Metal Gear Motor	Unit
Rated Voltage	6	V
Motor Speed	15000	RPM
No-Load Speed	310 rpm@6v	
No-Load Current	60	mA
Instant Torque	0.8	kg.cm
Weight	18	g

To power the engines a S7V8A step-up of the Pololu was used, since the motor shield does not supply power directly to the motor and the battery has a considerably low voltage. This step-up has been configured to provide 6 V to the motors, thus being able to obtain the maximum speed possible.

3.3.5 Sensors

Essential tasks performed by a mobile robot would not be possible without a constant interaction of the robot with the environment. Like any living being, the robot needs certain mechanisms to capture information from its environment and subsequently generate a specific behavior [64]. The mechanisms responsible for gathering the information from the environment are the sensors.

The sensors perform the signal capture from the environment, then the significant measurement information is extracted and based on them, the actions of the robot are planned. This process is explained in a simple way in the Figure 3.8, where 3 different phases are proposed. In the first phase "Sensing", the sensor collect the signal from the environment and convert it into computer data. The "planning" phase change the collected data into useful robot information like current position, speed and obstacle position. After, this information are associated with robot's commands [65]. Lastly, the set of commands is executed during the "action" phase [64].

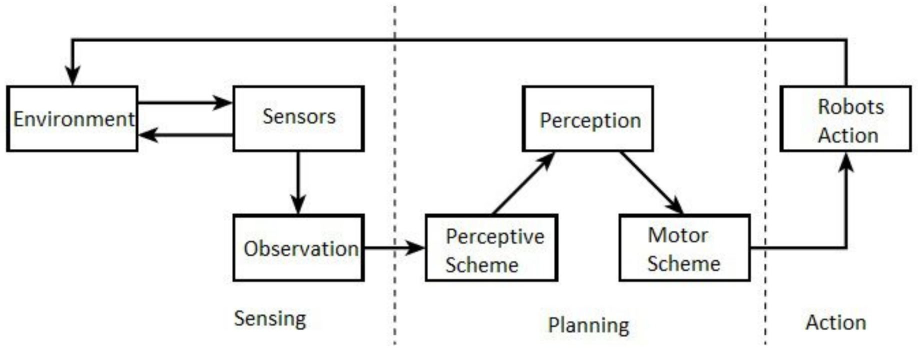


Figure 3.8: Simplified scheme of the process of interaction of the sensors with the environment.

In the current days there are several types of sensors, each one for a specific censoring. In this section, the odometer and the laser sensor used in this project are described.

3.3.5.1 Odometer

Odometry is the most widely used method for determining the speed and momentary position of a robot mobile, due to provide good short term accuracy, be cheap and allow very high sampling rates [66].

For this project optical encoders of 59 pulses per revolution were used. The optical encoder consists of a disk with several holes in its edge arranged at regular distances (Figure 3.9). This disk is connected to the motor, this way it rotates at the same speed as the wheel. When the wheel rotates, a beam of light passes through the holes of the disk, generating a square wave [65]. based on the analysis of the square wave frequency, the speed and distance traveled by the robot are estimated.

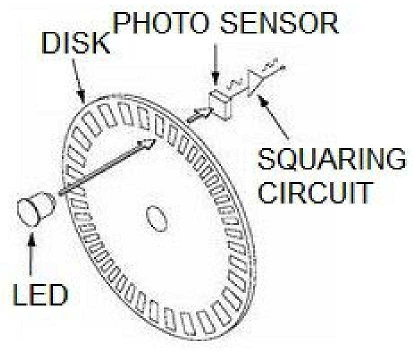


Figure 3.9: Optical encoder representation. Adapted from [66].

The principle of the odometry is the increment of motion information over time, which it is inaccurate due an unbounded accumulation of errors. Specifically, orientation errors will cause large lateral position errors, which increase proportionally with the distance traveled by the robot [66]. Typical odometry errors will become so large that the robot's internal position estimate is totally wrong after as little as 10 meters of travel [67].

This errors are caused by wheel slippage and some other more subtle causes, where wheel rotations do not translate proportionally into linear motion. According to [68] the errors can be categorized into one of two groups: systematic errors and non-systematic errors.

- **Non-systematic errors** are those errors that are caused by interaction of the robot with unpredictable features of the environment. For example, wheel slippage or bumps and cracks.

- **Systematic errors** are those resulting from kinematic imperfections of the robot, for example, unequal wheel diameters or uncertainty about the exact wheelbase.

To reduce these errors several researches propose methods for fusing odometric data with absolute position measurements.

3.3.5.2 Laser Distance Sensor

The laser distance sensor use a focuser light to measure distance to a target object. They detect solids objects independent of material, color and brightness. In this project, a VL53L0XV2 laser sensor was used (Figure 3.10). The VL53L0XV2 is a time-of-flight sensor that can report distances between 5 mm and 2 m with a resolution of 1 mm. This sensor uses a 940 nm laser to detect obstacles and communicate through I2C.

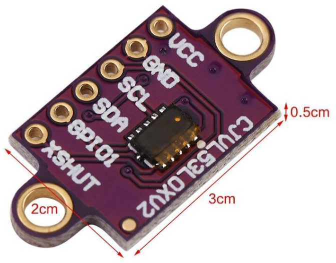


Figure 3.10: VL53L0XV2 sensor.

Time-of-flight technology is often used in long-range measurements. These sensors utilizes a transmitter diode to generate short pulses of infrared light which hits the surface of an object and is reflected into a receiver diode. The phases of the emitted and received light are compared and the distance between the sensor and the object is calculated [69], as presented in the Figure 3.11.

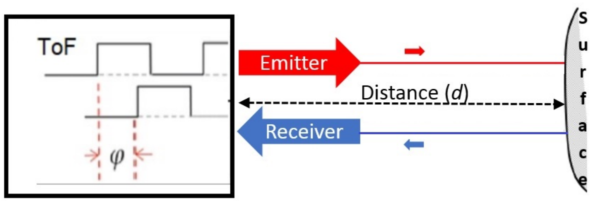


Figure 3.11: Time-of-flight technology model. Adapted from [70].

3.4 Robot Software

The robot software was developed directly on the Arduino platform using the C programming language. The developed code runs on the Wemos D1 mini integrated into the robot. This code is responsible for the odometry calculations, which together with the data obtained by the laser sensors and previous information on the map, estimate the position of the robot in the environment. It is also responsible for controlling the robot, mapping the environment, planning the path and controlling the speed of each engine.

Chapter 4

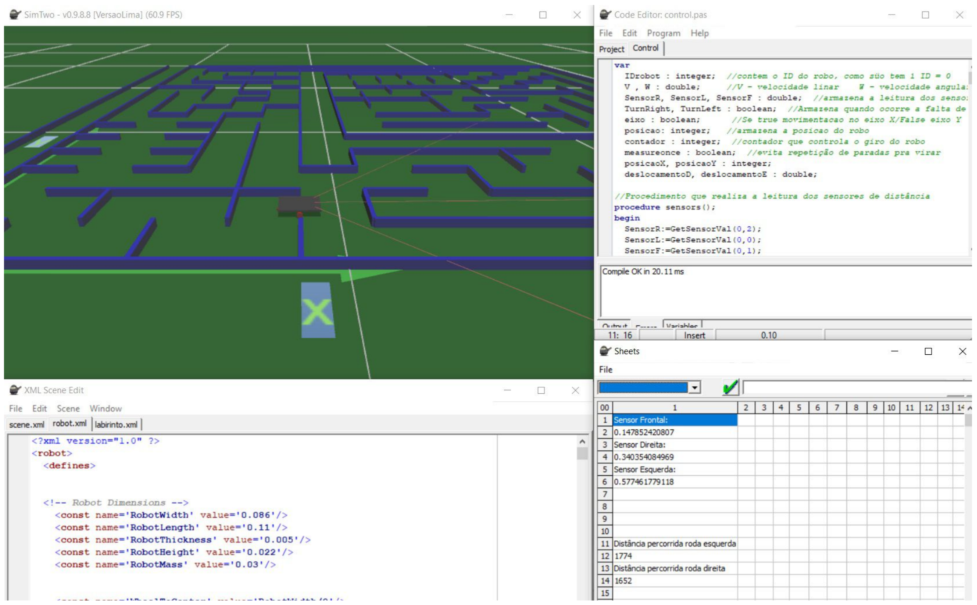
Simulation Model

The development of robots is a process that involves several areas of engineering, such as programming, electronics and mechanics, among others. In this sense, advanced technological materials and design methods are needed, which implies that the development of new robotic solutions is often an expensive practice [71]. Due to the financial factor and the possibility of evaluating the performance of algorithms before the actual implementation of the robot, the simulation has established itself as an important tool in the field of mobile robotics.

This chapter presents the developed simulation model, its implementation in SimTwo evidencing the dynamic model, the construction of the robot micromouse and the maze environment, and the development of a Hardware in the Loop (HIL) model.

4.1 SimTwo Simulator

SimTwo is a realistic simulation software suitable for the design and development of solutions for several types of robots. Its main purpose is the simulation of mobile robots that can have wheels or legs, although industrial robots, conveyor belts and lighter-than-air vehicles can also be defined. Basically any type of robot definable with rotating joints and/or wheels can be simulated [72]. Figure 4.1 presents the workspace of the simulation tool.



```
var
  IDrobot : integer;  //contem o ID do robo, como süo tem 1 ID = 0
  V , W : double;     //V - velocidade linar    W - velocidade angula
  SensorR, SensorL, SensorF : double;  //armazena a leitura dos senso
  TurnRight, TurnLeft : boolean;  //Armazena quando ocorre a falta de
  eixo : boolean;      //Se true movimentacao no eixo X/False eixo Y
  posicao: integer;   //armazena a posicao do robo
  contador : integer;  //contador que controla o giro do robo
  measureonce : boolean;  //evita repetição de paradas pra virar
  posicaoX, posicaoY : integer;
  deslocamentoD, deslocamentoE : double;

//Procedimento que realiza a leitura dos sensores de distância
procedure sensors();
begin
  SensorR:=GetSensorVal(0,2);
  SensorL:=GetSensorVal(0,0);
  SensorF:=GetSensorVal(0,1);
```

```
Compile OK in 20.11 ms
```

```
<?xml version="1.0" ?>
<robot>
  <defines>


  <!-- Robot Dimensions -->
    <const name='RobotWidth' value='0.086'/>
    <const name='RobotLength' value='0.11'/>
    <const name='RobotThickness' value='0.005'/>
    <const name='RobotHeight' value='0.022'/>
    <const name='RobotMass' value='0.03'/>
```

00	1	2	3	4	5	6	7	8	9	10	11	12	13	14
1	Sensor Frontal:													
2	0.147852420807													
3	Sensor Direita:													
4	0.340354084969													
5	Sensor Esquerda:													
6	0.577461779118													
7														
8														
9														
10														
11	Distância percorrida roda esquerda													
12	1774													
13	Distância percorrida roda direita													
14	1652													
15														

Figure 4.1: SimTwo micromouse simulation interface.

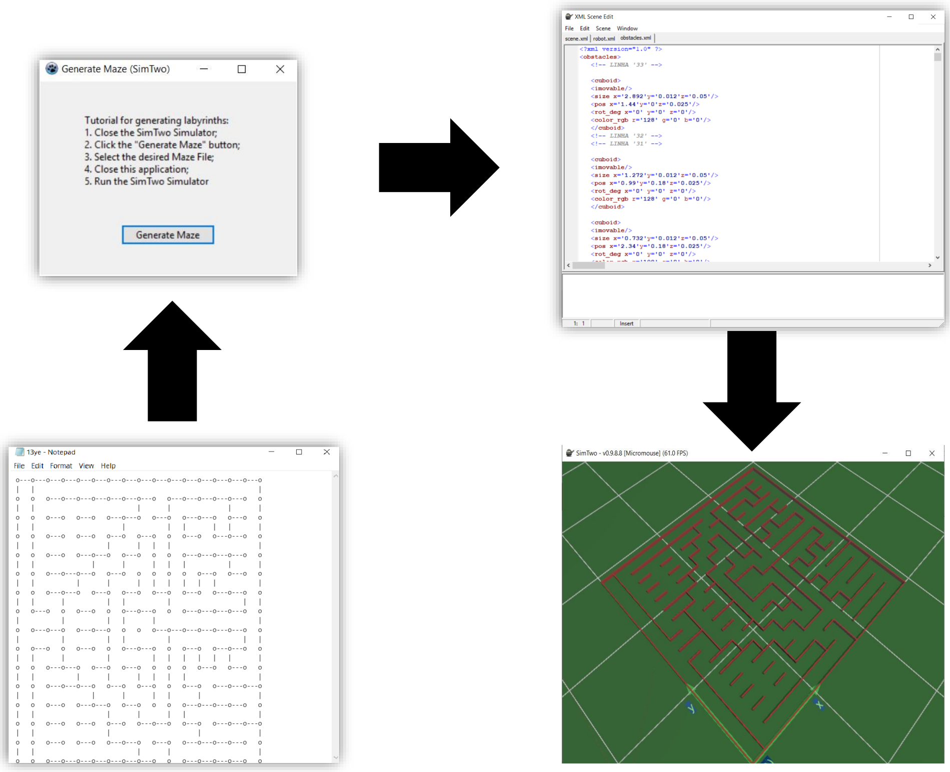
The simulator provides a platform where several robots can be simulated at the same time. Each robot is defined by various solids (cuboid, cylinder and sphere) interconnected through joints (slider, socket and hinge) and sensors that detect information from the robot and the environment. The dynamics realism is obtained by simulating each body and electric motor numerically using its physical characteristics: shape, mass and moments of inertia, surface friction and elasticity [11].

This application is based on a multiple document interface where the scenario and the robot's body and dynamics are implemented on XML language, meanwhile the robot's control is implemented on Pascal programming language.

4.1.1 Maze Generator

Based on the necessity of validate and test the developed micromouse algorithms in the most varied scenarios, a maze generator was implemented. Such generator has the capacity to represent more than 450 different classic labyrinths (with valid start and goal), in a simple and fast way to change between them. Those labyrinths reproduce the competition environment with greater fidelity, having the same characteristics of a real competitive maze, as described in section 1.1.

Originally, the labyrinths are encoded in a text file to facilitate their visualization (credits to [73]). The developed maze generator extracts the information from the text file and encodes it into an "obstacle" file in XML which is interpreted by the SimTwo simulator. The maze generation is shown in Figure 4.2, where the maze in text file format is selected for the conversion and the resultant XML file is produced. After finishing the maze generation it is possible to observe the 3D simulation environment with the selected maze.


Generate Maze (SimTwo)
Generate Maze
XML Scene Edit
File Edit Scene Window
scene.xml robot.xml obstacles.xml
1: 1 Insert
13ye - Notepad
File Edit Format View Help
SimTwo - v0.9.8.8 [Micromouse] (61.0 FPS)

Tutorial for generating labyrinths:
1. Close the SimTwo Simulator;
2. Click the "Generate Maze" button;
3. Select the desired Maze File;
4. Close this application;
5. Run the SimTwo Simulator

```
<?xml version="1.0" ?>
<obstacles>
  <!-- LINHA '33' -->

  <cuboid>
  <imovable/>
  <size x='2.892'y='0.012'z='0.05'/>
  <pos x='1.44'y='0'z='0.025'/>
  <rot_deg x='0' y='0' z='0'/>
  <color_rgb r='128' g='0' b='0'/>
  </cuboid>
  <!-- LINHA '32' -->
  <!-- LINHA '31' -->

  <cuboid>
  <imovable/>
  <size x='1.272'y='0.012'z='0.05'/>
  <pos x='0.99'y='0.18'z='0.025'/>
  <rot_deg x='0' y='0' z='0'/>
  <color_rgb r='128' g='0' b='0'/>
  </cuboid>

  <cuboid>
  <imovable/>
  <size x='0.732'y='0.012'z='0.05'/>
  <pos x='2.34'y='0.18'z='0.025'/>
  <rot_deg x='0' y='0' z='0'/>
```

Figure 4.2: Maze conversion process.

4.1.2 Robot Configuration

For the simulator development, SimTwo uses an Open Dynamic Engine. This is an open source library of simulation functions that support movement and connection of rigid bodies, rotational inertia and treatment of collisions. Thus, in the development of the simulation world it is only necessary to introduce a few object's information, like dimension, location, mass and connections [74].

The simulated micromouse robot was assembled with the same dimensions as of the real one, presented at the Table 3.1. The model is a combination of solids (cuboids and cylinders) and shells, elements without mass that do not modified the robot physical properties but are an essential part of the collisions simulation. Table 4.1 show the ID of each object along with their mass, size and position in relation to the reference point $(0, 0, 0)$ of the simulator.

Robot	Description	Mass (kg)	Position (m)			Size (m)		
ID			X	Y	Z	X	Y	Z
1	Base Plate	0.8	0	0	0.0195	0.11	0.086	0.005
2	Wheel Left	0.15	0	0.043	0.08	0.008	0.008	0.008
3	Wheel Right	0.15	0	-0.043	0.08	0.008	0.008	0.008
4	Caster Wheel	0.1	-0.055	0	0.004	0.004	0.004	0.008
5	Caster Pole	0.05	-0.045	0	0.017	0.004	0	0.01292
6	Bracket Left	0	0	0.031	0.016	0.012	0.012	0.007
7	Bracket Right	0	0	-0.031	0.016	0.012	0.012	0.007
8	Motor Sheel Right	0	0	0.033	0.008	0.002	0.002	0.012
9	Motor Sheel Left	0	0	-0.033	0.008	0.002	0.002	0.012

Table 4.1: Micromouse simulated structure.

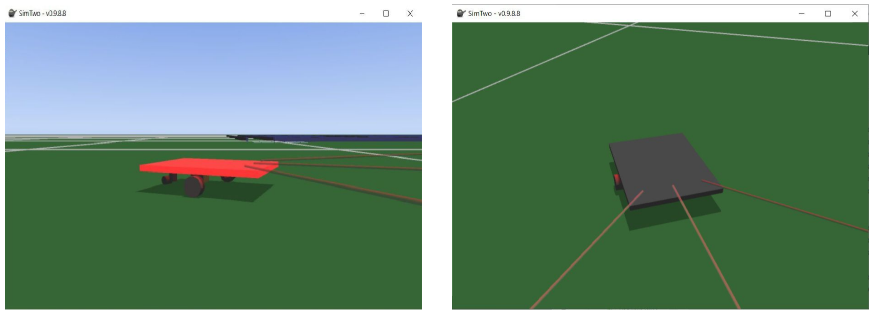
To connect the parts hinge joints were used. This type of joints allows two objects to move between them through a single axis. The connected parts were: the right and left wheel on the base plate, the caster pole on the base plate and the caster wheel on the caster pole.

In order to navigate and map the environment three distance sensors were used. They are located in the frontal part of the base plate with angles of 45, 90 and 135 degrees. The sensors length, angle and position in relation to the reference point of the simulator

Table 4.2: Position of the sensors in Simtwo simulation.

Sensor	Position (m)			Angle (degrees)	Length (m)
	X	Y	Z		
Right	0.04	0.071	0.004	45	0.5
Left	0.04	-0.071	0.004	90	0.5
Middle	0.04	0	0.004	135	0.5

are presented in Table 4.2. The resulting robot is presented in Figure 4.3.



(a) (b)

Figure 4.3: Simulated micromouse robot : (a) Lateral view. (b) Top view.

4.1.3 Robot Control

The robot's control is defined in the "code editor". This editor offers an IDE for high-level programming based in Pascal language. The control script is divided into two main procedures, Initialize and Control. The Initialize procedure is executed once the code starts setting the initial configuration of the variables. Meanwhile the control procedure repeats itself every 40 ms executing the functions responsible for the control of the robot. The resulting robot movements obtained of such functions can be visualized in the main window, Figure 4.1.

The robot's locomotion is based on sensing the environment, self-locating and setting the speed for each motor. SimTwo offers a library of functions that provide the information required for such actions.

Initially, to navigate without collision and map the labyrinth environment, the micromouse requires a constant interaction with the environment. Such interaction is performed by the three sensors that measure the distance between the robot and the walls. These measurements are obtained using the Simtwo *GetSensorValue(IDrobot, IDsensor)* function that returns the distance of the obstacle in millimeters. Based on such values, a collision avoidance algorithm is implemented to keep the robot at the center of the cell, this algorithm is presented in the section 5.3.1.

The robot's movements is based on the motors actuation. To control the speed and direction of the robot two variables are implemented, V and W. The V variable stores the linear velocity of the wheels, controlling the final speed of the robot, while the variable W keep the angular velocity, controlling the direction of the robot. The angular velocity is related to the distances measured by the collision avoidance algorithm, meanwhile the linear velocity is either pre-set or related to the proximity between the robot and frontal wall. To define the final speed of the robot the Algorithm 1 is applied.

Algorithm 1 Motor's Speed Control

```
1: procedure SpeedControl($V, W$)                          ▷ angular and linear speed
2:     $FinalSpeedRight \leftarrow V - W$
3:     $FinalSpeedLeft \leftarrow V + W$
4:     $SetAxisSpeedRef(IDrobot, IDmotor, FinalSpeedRight)$
5:     $SetAxisSpeedRef(IDrobot, IDmotor, FinalSpeedLeft)$
```

In order to map the environment and perform the planned path the robot need the capacity of self-locate in the maze. This is implemented utilizing the encoders present in the motor. The measured number of pulses per cycle (40 ms) can be obtained using the SimTwo command *GetAxisOdo(IDrobot, IDmotor)*, based on this value is possible to obtain the current speed and position of the robot.

4.1.4 Results Presentation

SimTwo present two main tools for real time evaluation of the algorithms, the chart and the spreadsheet windows. The chart allows to plot the current value of the variables utilized in the control, such as sensors distance, robot's position, and motor speed. By this, incoherent values are easily found and corrected.

In the spreadsheet window it is possible to customize the value of the cells and associate them with variables present on the control. It is also possible to define the cells as buttons for specific operations, enabling the remote control of the simulation. This window is a great asset in the micromouse project, since it is possible to configure the cells as grid cells and generate a 2D map of the maze according to the movement of the robot, as shown in Figure 4.4.

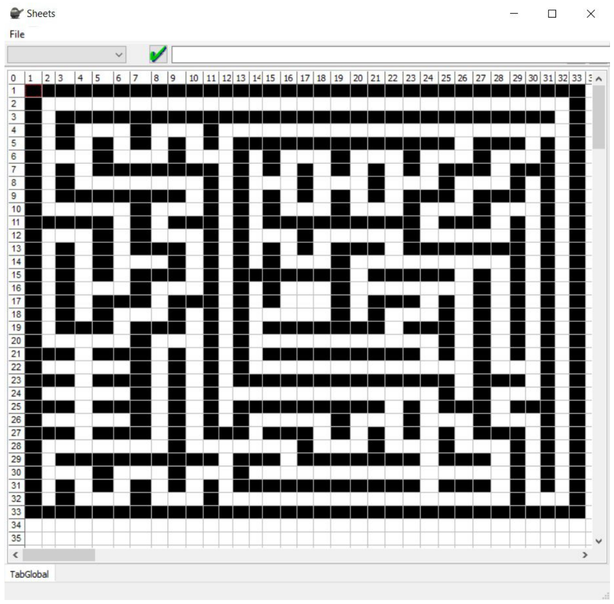


Figure 4.4: Maze presentation in the Spreedsheet window.

4.2 Hardware in the Loop

The technological advances that occurred in the last decades have increased the use of robots and embedded systems in everyday life and especially in industry. However, before these new systems are definitely available in the market, several tests must be carried in order to validate the operation of the system and ensure user safety. Many of these tests may fail and compromise operators and systems. The simulators have emerged to avoid these risks and speed up the development of the system. In this context, the HIL becomes a powerful tool to be used before the actual tests, replacing the electromechanical systems with a dynamic 3D simulator.

HIL is a real-time simulation for control systems, in which the actual processor and other control systems are combined with the dynamic simulator to simulate the actual system [75]. These systems lies between the real world and the simulation, as shown in Figure 4.5, where it can be observed that the HIL presents more precision than the simulation and it is cheaper than implement the tests in a real system.

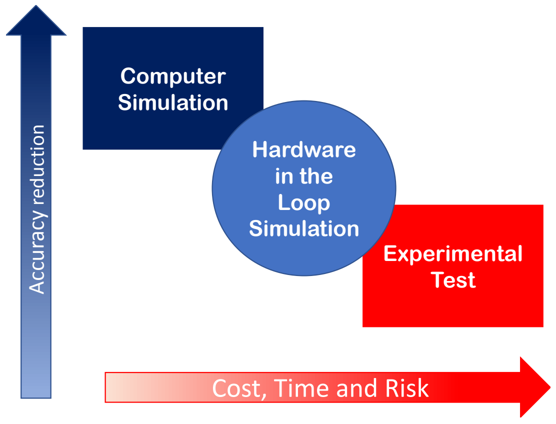


Figure 4.5: Correlation of price, time, risk and precision between simulation and real systems. (adapted from [76]).

The HIL systems have been widely applied, since they allow to find software and hardware errors at the beginning of the development process, and their results are very close to the real ones. The main areas that have been developing and adopting these systems are the aeronautical industries [77], the automotive industries [78], [79], power systems [80] and robotics areas [81], [82].

In this project, a HIL was applied using the SimTwo simulator and a Wemos D1 mini. Real-time communication between these devices was established through the USB port (Serial communication). In this communication, the simulator provides sensor data (right, left and front distance sensor and motor encoders) to the hardware-based controller, which processes this data and controls the actions of the simulated robot (right and left motors), as shown in Figure 4.6.

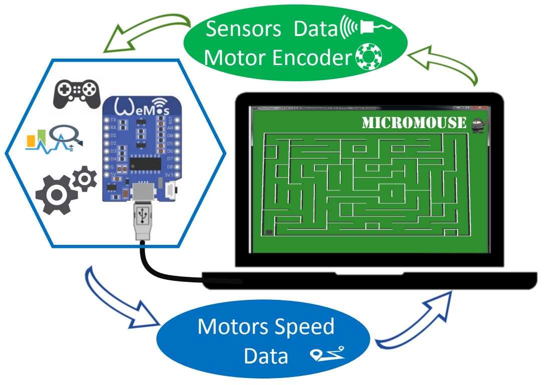


Figure 4.6: Architecture HIL for Micromouse SimTwo simulator.

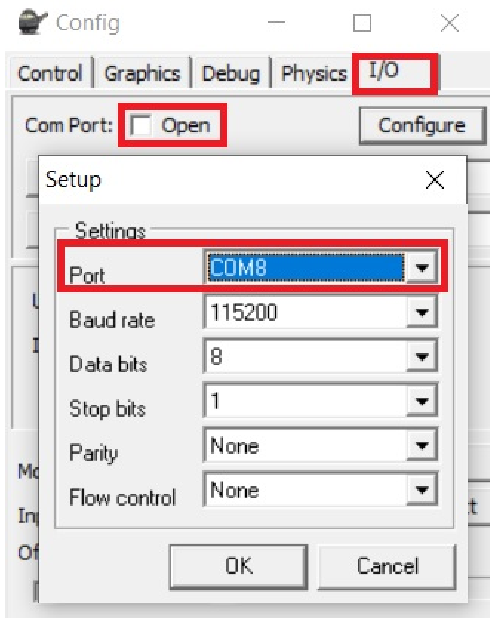
With this tool it is possible to develop the micromouse algorithm considering the limitations presented by the controller, memory and processing time limitations, and to verify the performances obtained in several different labyrinths, using the maze generator, without having to assemble them. This way it is possible to verify the efficiency of the algorithms developed in the most diverse scenarios, with a considerably low cost and results very close to those of the real implementation.

A simple wall-follower implementation, based on the robot presented in Section 3 is available on [83]. To test it the following steps must be performed:

1. Compile the code in the controller (Arduino IDE) and then close all dependencies of the Serial port (Monitor Serial);

2. On the SimTwo, in the $Config->I/O$ tab (Figure 4.7), configure the Serial port with a BaudRate of 115200 [bps] and select the COM port corresponding to the Arduino IDE, and then open the communication. To finish this step, open the communication by selecting the "open" checkbox;

3. On the *Editor* tab, compile the code ($Ctrl + F9$) and then execute pressing ($F9$). After this step the HIL tool will be in operation.



Port	COM8
Baud rate	115200
Data bits	8
Stop bits	1
Parity	None
Flow control	None

Figure 4.7: SimTwo HIL configuration.

Chapter 5

Localization, Mapping, Path planning and Control

This chapter will present the means for a mobile robot to be able to sweep an indoor environment. Thus, three major research areas of mobile robotics will be discussed: localization, path planning and mobile robot control.

5.1 Maze representation

The micromouse maze presents well defined structure, being composed by 16 $\times$ 16 multiple square units (cells) of 18 cm $\times$ 18 cm. Each wall of the labyrinth is 5 cm high and 1.2 cm wide, leaving 16.8 cm of free space between them [7]. This maze is surround by an outer wall with the same dimensions.

Due to the format of the obstacles, the complexity of the maze and the constant ambiguities present on the labyrinth, a cell decomposition method was implemented. To represent the entire maze a binary matrix of 33 $\times$ 33 were implemented. This matrix have four different cells size according to its axis position:

- Cells in odd columns are 16.8 cm high and 1.2 cm wide;

- Cells in odd lines are 1.2 cm in height and 16.8 cm in width;

- Cells in odd rows and columns are 1.2 cm in height and width;

- Cells in other positions are 16,8 cm in height and width.

This configuration allows the perfect representation of the maze into a matrix maintaining the free cells and obstacles original size (Figure 5.1). To identify the contents of the cell, obstacle or free path, an identifier has been placed in each cell. If this identifier displays the value one, the cell contains an obstacle, otherwise the cell is a free path.

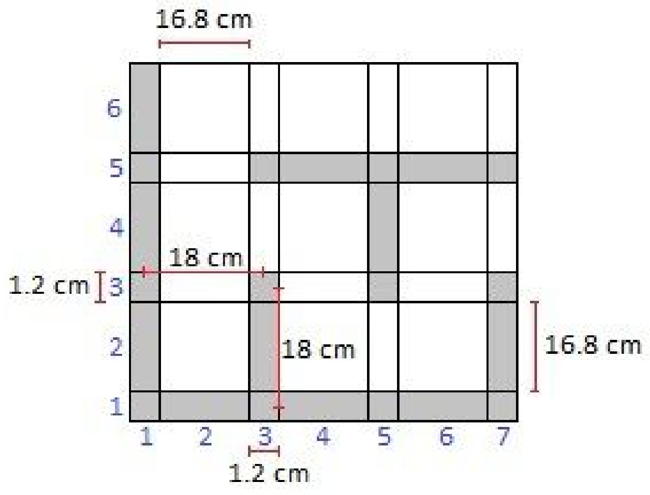


Figure 5.1: Partial representation of the maze matrix (white - free cell, grey - obstacle).

5.2 Localization

The micromouse contest provides an initial position for the robot within the maze, this position is located in one of the four corners of the labyrinth [7]. This initial position is defined as the reference point for the robot location system. The displacement of the robot in relation to the reference point is measured using the encoders present on both wheels, according to

$$Displacement = \left(\frac{Wheel_{circumference} \cdot Cont_{Right}}{PR} + \frac{Wheel_{circumference} \cdot Cont_{Left}}{PR} \right) \cdot \frac{1}{2} \tag{5.1}$$

$$Position = Position + Displacement \tag{5.2}$$

where, PR are the number of peaks per revolution given by the encoder and $Cont_{Right}$ and $Cont_{Left}$ is the current number of measured peaks, these values are reset every time the measurement is performed.

Due to the need to map the environment two-dimensionally, a two-variable record was implemented to maintain the location of the robot inside the maze. This record contains the variables X and Y, which stores the displacement (in millimeters) of the center of the robot in relation to the axes of the labyrinth. The variables X and Y are initiated, respectively, with values of 90 and 70 which correspond to the initial position of the center of the robot, due to the fact that the robot starts the competition in the center of the reference cell and with the back against the wall of the labyrinth.

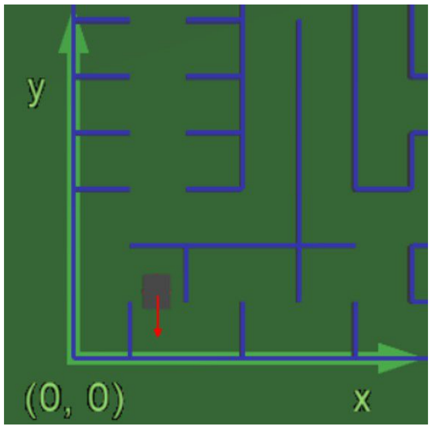


Figure 5.2: Partial representation of the maze and its axes.

The robot starts by moving in the Y-axis of the labyrinth, consequently the distance traveled by the robot is added to the value stored in the variable Y. The exchange between the displacement variables occurs whenever the robot makes a curve. However, if the robot is moving in a direction opposite to the axes, as in Figure 5.2, this movement must be subtracted from the value stored in the variable corresponding to the axis. To maintain control of the direction in which the robot is moving, a flag variable has been created. Based on the information of such variable is defined whether the displacement will be added or subtracted from the stored value.

In order to obtain a reliable mapping and to perform the planned path at the lowest possible speed without collision, it is necessary to have the precise location of the robot in real time. The use of only the odometric sensors for location does not provide this required precision. As shown in Section 5, the odometry presents cumulative errors, in this project most of the odometric errors are due inaccuracies in wheel dimensions, lack of balance, bumps and skidding. Using the distance sensors, the information obtained by the odometry and a previous knowledge of the dimensions of the maze, two algorithms were developed to reduce the imprecision of the robot's location. Such corrections are made when:

- There is a side wall near the current position of the robot. In this case, using the distance obtained by the laser sensor, the variable related to the axis that the robot is not currently moving is updated according to

$$Position = Current_{Position} + ((Position \quad DIV \quad Cell_{Size}) \cdot Cell_{Size}) \tag{5.3}$$

  where the equation of the current position in the cell varies between

$$Current_{Position} = (Sensor_{Lateral} \cdot \cos 45^{\circ}) \tag{5.4}$$

  and

$$Current_{Position} = Cell_{Size} - (Sensor_{Lateral} \cdot \cos 45^{\circ}) \tag{5.5}$$

  according to the axis and direction of movement and the side of the measured sensor.

- There is a wall in front of the robot before it realizes a curve. The variable referring to the axis that the robot was moving is updated according to

$$Position = Current_{Position} + ((Position \quad DIV \quad Cell_{Size}) \cdot Cell_{Size}) \tag{5.6}$$

where the equation of the current position in the cell varies between

$$Current_{Position} = Cell_{Size} - Sensor_{Frontal} \tag{5.7}$$

and

$$Current_{Position} = Sensor_{Frontal} \tag{5.8}$$

according to the direction of movement. To avoid errors, this update is made in the short time the robot is stopped before making the curve.

5.3 Control

The robot's control is obtained by the configuration of the speed to be applied in each motor. The set of such velocities defines the robot's final speed and the direction in which it moves. Two variables were used to control these characteristics. Such variables configure the angular and linear velocity to be performed by the robot. The speed applied to each motor is defined according to

$$V_{RightWheel} = V - W \tag{5.9}$$

and

$$V_{LeftWheel} = V + W \tag{5.10}$$

where $V_{RightWheel}$ and $V_{LeftWheel}$ are the speed applied in each motor.

The linear velocity is controlled according to the activity being performed. During the mapping phase, a low speed is used to increase the accuracy of the generated map. This speed is increased for the evaluation run, aiming to reduce the time needed for the robot to reach the center of the maze. In order to reduce bumps and skids, especially in the evaluation run where the robot speed is greater, an acceleration and deceleration algorithm has been implemented. The acceleration algorithm constantly increases the speed of the robot until it reaches the maximum speed allowed in the task being executed. Meanwhile, the deceleration algorithm constantly reduces the speed of the robot until it reaches the point where the curve will be made.

The angular velocity is specially applied on two occasions, when executing a curve/rotation, and correcting the movement of the robot. In these cases, different speeds or direction are applied to the wheels, thus changing the direction in which the robot is moving.

5.3.1 Centralization Algorithm

A centralization algorithm has been implemented to increase the accuracy of the generated map, ease the robot's control and avoid collisions during its movements. This algorithm uses the distance of the walls obtained by the laser sensors and the previous knowledge of the size of the cells to keep the robot centered around the current unused axis.

The centralization algorithm analyzes the current situation of the robot in the maze and returns the angular velocity to be applied in the motors in order to correct the robot's positioning. Taking into account the possible variations of the positions of the walls in relation to the robot, this algorithm accomplishes the correction of the robot's position in four configurations common in the mazes. The four different cases are presented below.

5.3.1.1 Both Close Lateral Walls or a Large Frontal Wall

On these occasions, shown in the Figure 5.3, it is possible to make a comparison of the values obtained by the lateral sensors. Based on the difference between these values an angular velocity is defined to correct the positioning of the robot, as shown in

$$W = K \cdot (Sensor_{Right} - Sensor_{Left}) \tag{5.11}$$

where K is a proportional gain, obtained in tests, to accelerate the centralization process.

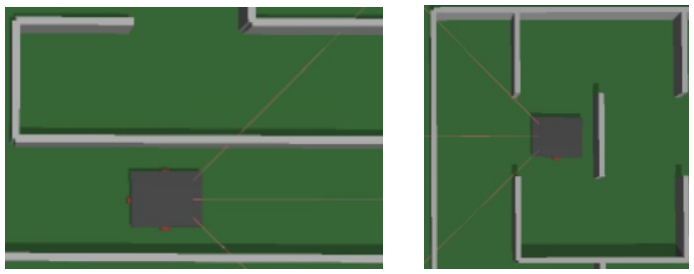

(a) (b)

Figure 5.3: Environment situations: (a) Both close lateral walls. (b) Large frontal wall.

5.3.1.2 One of the Close Lateral Walls

In these cases the side walls are at quite different distances (Figure 5.4), not being possible to make a comparison. In such cases triangulation is necessary. Using the known angle of the sensor (45 degrees) and the distance measured by the sensor, the current distance between the wall and the robot is calculated. This distance is then compared to the distance required for the robot to be in the center of the cell and the angular velocity is adjusted. The complete calculation of this variance is presented in

$$W = K \cdot \left(SensorDist - \left(\frac{x}{cos(45)} \right) \right) \tag{5.12}$$

where x is the distance between the robot's sensor and the wall (41.5 mm), K is a proportional gain, obtained in tests, and the W signal varies according to the measured sensor side.

5.3.1.3 One of the Sidewalls a Cell Away

Very similar to having a close lateral walls and in these cases a triangulation is also performed. However, the distance that is compared is the sum of the length of a cell

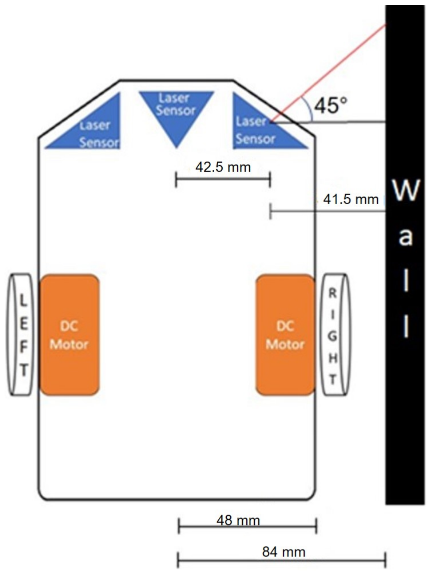


Figure 5.4: A single wall close.

with the distance required for the robot to be in the center of the cell. In this case, the Equation (5.12) is applied by varying the value of x to 215.5 mm.

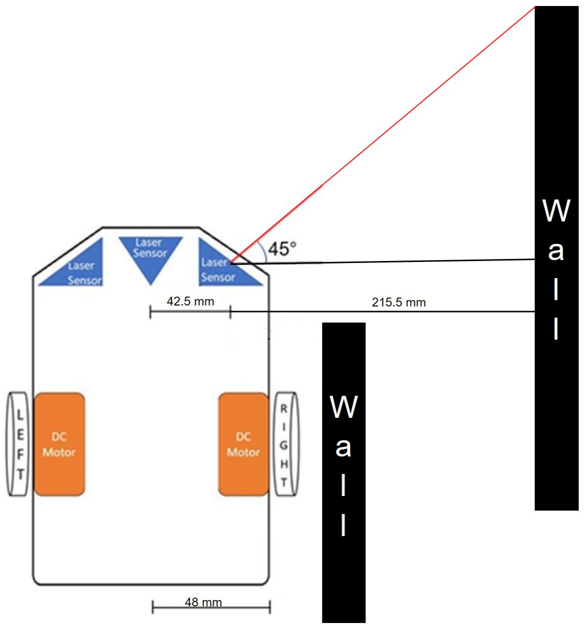


Figure 5.5: A single wall a Cell Away.

5.3.1.4 Other cases

If the environment around the robot does not fit in the situations mentioned above, it is not possible to perform the robot centralization. In these cases the angular velocity is set to zero and the robot moves in a straight line. The centralization will be performed when the robot moves to an environment similar to previous ones.

To increase stability and accelerate the centralization process, the value obtained by the centralization algorithm is multiplied to a proportional gain.

5.3.2 Curves

To ease the location control and reduce the required correction in the robot's direction to keep it centralized, the curves/rotations are performed when the robot is as close as possible to the center of the cells. To perform the curves and rotations, the linear velocity is set as zero and an angular velocity is defined. This way, the wheels rotate at the same speed, but in opposite directions, rotating the robot without changing the central axis position. Therefore, its not required to update the robot's position during a curve or rotation.

To control the angle of rotation performed by the robot, the odometry of the external wheel is used. This angle is obtained by

$$\theta_{Rotation} = \theta_{Rotation} + Tg^{-1}\left(\frac{\frac{Wheel_{circumference} \cdot Cont}{PR}}{b}\right) \tag{5.13}$$

where PR is the number of peaks per revolution given by the encoder, $Cont$ is the current number of measured peaks and b is the distance between the center of the robot and the wheel, these values are reset every time the measurement is performed. The maximum value of $\theta_{Rotation}$ varies according to the command to be executed, if it is a curve the robot turns 90 degrees and in case of a rotation it rotates 180 degrees. Upon reaching the desired angle, the angular velocity is set to zero and the linear velocity is defined according to the task being performed by the robot.

5.4 Search Algorithms

Once the mapping is complete, it is necessary to search for the best path through it. This task is performed by search algorithms. These algorithms systematically search a graph with the goal of finding a specific cell [33]. During this research, a record of the movements that the algorithm performed to find the goal is kept. When the desired cell is found, the moves are checked to get the best path. The generated path is a sequence of cells that the robot must traverse to reach the goal. This section presents an optimization to the A* search algorithm seeking to reduce the processing time while maintaining the path quality.

5.4.1 A* Modified

There are several implementation details that can significantly affect the performance of an A* algorithm. This modified A* implementation implements two modifications in order to reduce processing time. The first modification is in the form that the cells to be visited are stored. This version uses the binary heap data structure to store this nodes. A binary heap is a data structure created by using a binary tree that can be constructed by successive insertions.

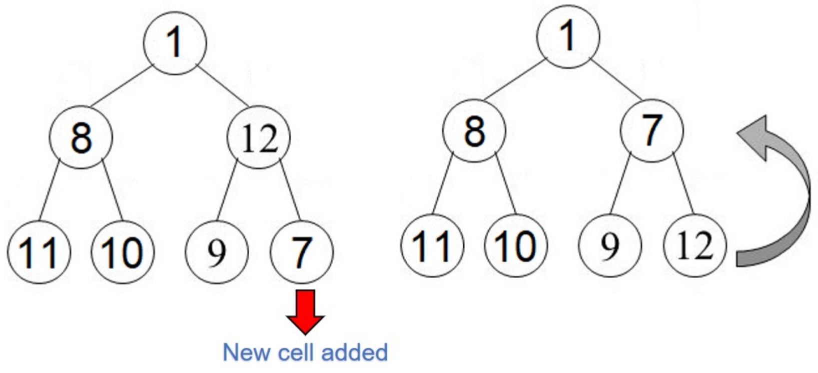


Figure 5.6: Ordering a new cell in a binary heap.

One of the biggest expenses in the processing time of algorithm A* is to order the cells to be visited according to their cost. In the binary heap this duration is reduced, since positioning the cell accordingly to its cost just requires to verify the costs of its parents cells (Figure 5.6), no longer is it necessary to verify the majority of the cells in the list.

The second modification is the addition of the k value into the heuristic equation. This allows to adjust the search space, with $k = 1$ there is the guarantee that the final solution will be optimal, however using a higher value for k the search space is reduced and the solution found can be suboptimal. The modified heuristic equation

$$h(x, y) = k\sqrt{(x - x_t)^2 + (y - y_t)^2} \tag{5.14}$$

where the $(x,\ y)$ are the coordinates of the current position and $(x_t,\ y_t)$ are the goal coordinates.

The advantage of reducing the search space is that the computing time is also reduced. Sometimes the trade off can be advantageous if the computing time gains are significant and the path length stays near the optimal. This is very important in real time scenarios.

5.5 Mapping

The mapping process occurs as the robot moves through the unknown maze, therefore, an algorithm of motion in an unknown environment is required. In the development of this project two algorithms of movement in unknown environment were implemented. These algorithms are presented below:

- Wall Follower

  The most common algorithm for a maze solver robot is the wall follower algorithm, also called left hand right hand rules. In this method the robot will decide its direction by following the left or right wall. Whenever the robot arrives at a junction, it will detect the opening walls and select its direction, giving priority to the selected wall [84], in the case developed the right wall was selected. This selection occurs according to the following algorithm.

1. Sense the right wall.
2. If the right wall is not present, turn 90 degrees right and return to step 1.
3. Sense the front wall.
4. If the front wall is not present, move straight and return to step 1.
5. Sense the left wall.
6. If the left wall is not present, turn 90 degrees left, else rotate 180 degrees.
7. Return to step 1.

Taking the walls as a guide, this strategy is able to make the robot reach the goal of the maze without actually solving it. However, this algorithm is not an efficient method to solve a maze since the wall follower algorithm is not capable to solve labyrinths with a closed loop region [85].

- Wall Follower with Repositioning Algorithm

  In order to avoid the path repetition problem found in the Wall Follower algorithm, a repositioning algorithm was developed. This algorithm uses the base of the wall follower but with a variation, when going through a repeated path a search algorithm is used to find the nearest unexplored entrance. When this entry is found, the search algorithm returns the cells that must be traversed by the robot to reposition itself. This path is followed by the robot until it reaches the destination, then the mapping starts again using the wall follower algorithm.

  To perform this repositioning algorithm, two conditions are required. During the mapping, the unexplored entries must be marked with a different value to indicate the position to be found by the search algorithm. The second condition is the use of a search algorithm with low processing time, since the generated path does not have to be as short as possible and the time needed to process this path can cause collisions due to the robot continuing to move in this period. Therefore, the A* modified algorithm with a high K value was applied.

Initially an unknown value is assigned to all the cells in the maze. As the robot moves through the labyrinth these cells are updated according to their content, if there is an obstacle, the cell receives a value of 1, otherwise it receives a value of 0. This update happens when the robot passes the center of the cell. Using the location of the robot in the X and Y axes and the measures obtained from the laser sensors, the values of the frontal and diagonal cells to the robot are defined. An example of mapping is shown in Figure 5.7, where the robot's current position is shown in red, obstacles are displayed in black, free paths are white, and non visited cells are blue.

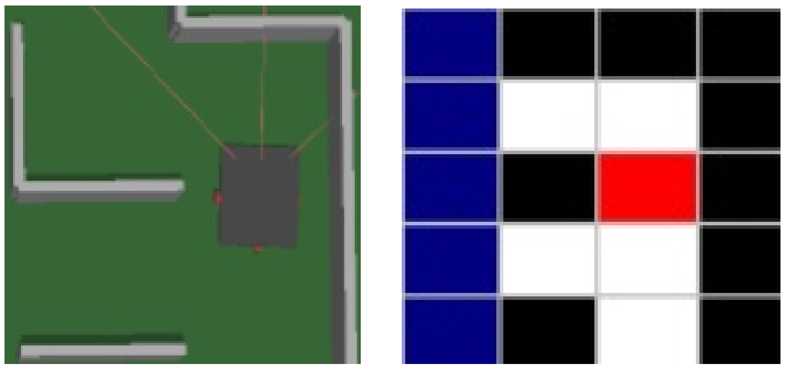

(a) (b)

Figure 5.7: Mapping process: (a)Robot's current position. (b) Generated map.

5.6 Path Planning and Evaluation Run

Based on the previously obtained map, a path planner is applied. This path planner returns a cell path that must be visited for the robot to reach the target as quickly as possible. In order to facilitate the robot's control during the evaluative run, this sequence of cells are analyzed and transformed into simple commands for the robot. Comparing the values of the axes of the cell with its predecessor, it is possible to obtain the command that must be executed for the robot to continue on the rail.

From the analysis of the cells it is obtained a vector with commands: front, right and left. It starts with the robot moving frontally. When the robot changes cell, the first command of the vector is executed and this vector is updated. This is repeated until the goal is achieved. It should be noted that only 90 degree turns were considered in the evaluation run.

Chapter 6

Results

This section is dedicated to the presentation of the results related to the localization, mapping, planning and control activities, using the methodologies presented in the previous chapters. In order to perform this evaluation, experiments were carried out in the SimTwo simulation environment, as well as practical experiments with the real robot.

Initially, the cumulative error generated by the odometry in the location of the robot is introduced. Along with this introduction, the results obtained by the correction of the location using the distance sensors are presented. Then, the methods developed for the robot to move in an unknown environment and map it are compared. After, a comparison of different path planning methods is realized, verifying the quality of the path generated, the number of visited cells, and the processing time required to obtain this path. Finally, the results obtained in the evaluation run are presented.

6.1 Simulation

The SimTwo simulator provides a realistic environment allowing the configuration of several robotics and environmental characteristics, such as the motor settings and environment friction, letting the verification and correction of the non-idealities errors present in the implementation of the real robot. The simulator also provides tools for real-time monitoring of measures and robot configurations, enabling a quick detection and localization of errors present in the code.

The simulation environment also allows a better comparison of different methodologies, since it is possible to obtain the results of their applications in the most varied mazes without having to assemble them. This section presents the results obtained during the development of the robot micromouse in simulated environment.

6.1.1 Localization

The main error present on the real development of the micromouse is due to non-systematic errors that occur on the odometer measurements caused especially by wheel slippage and bumps. To check this error and minimize it, the simulator provides the position of the robot in real time, making it possible to compare the data of the odometry and the real position/velocity. To perform this comparison, the path of the Figure 6.1 was traveled by the robot.

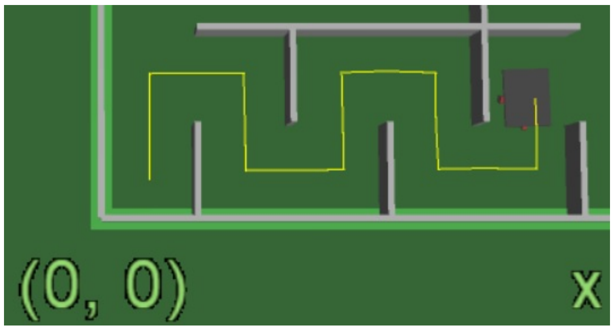


Figure 6.1: Traveled route.

Figure 6.2 presents the comparison between the odometry and the real positioning of the robot in Figure 6.1. It is observed that the difference starts small and it increases as the robot moves, especially when performing curves due to the need to decelerate, accelerate and correct its positioning. This increase occurs due to the error being cumulative, the small errors are adding up, which causes great variations after some displacement.

The micromouse robot requires a precise location to be able to map correctly and go through a selected path without colliding, so using only the odometry to locate has proven to be inefficient. In order to increase the accuracy of the localization system, the

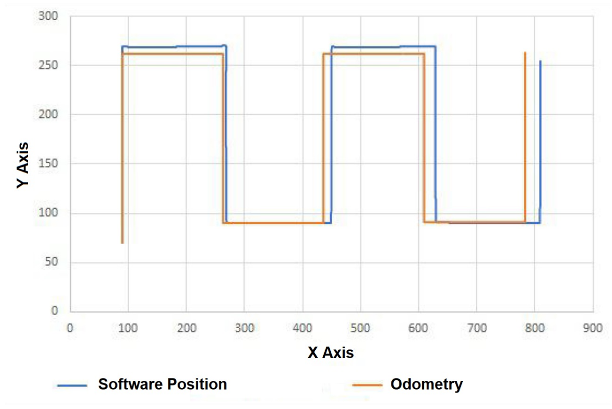


Figure 6.2: Comparative between odometry and real robot movimentation.

method presented in the section 5.2 has been implemented. This method uses previous knowledge of the dimensions of the maze, the measurements obtained by the distance sensors and odometry to avoid that the localization error accumulates as the robot moves. This method still shows a small variation between the measured position and the current position, but this variation remains constantly small throughout the trajectory and can therefore be neglected.

6.1.2 Mapping

The mapping of an unknown environment is one of the most important step in the software development of a robot for micromouse competition, since the path planning and the evaluation run depend on the quality of the map generated and the amount of environment explored. The micromouse competition has a maximum time for the execution of all required tasks being 10 minutes. Therefore, a time for each task is defined. In order to allow plenty of time for the evaluation race, a mapping time of 7 minutes was established.

This section presents and compares the maps obtained with different locomotion algorithms for unknown environments. These algorithms are evaluated in several criteria, especially the percentage of the map explored and if the center of the labyrinth was found. To be able to evaluate the performance of the algorithms, three maps used in past competitions of the micromouse were selected. These maps are shown in the Figure 6.3. The locomotion algorithms are presented according to its complexity.

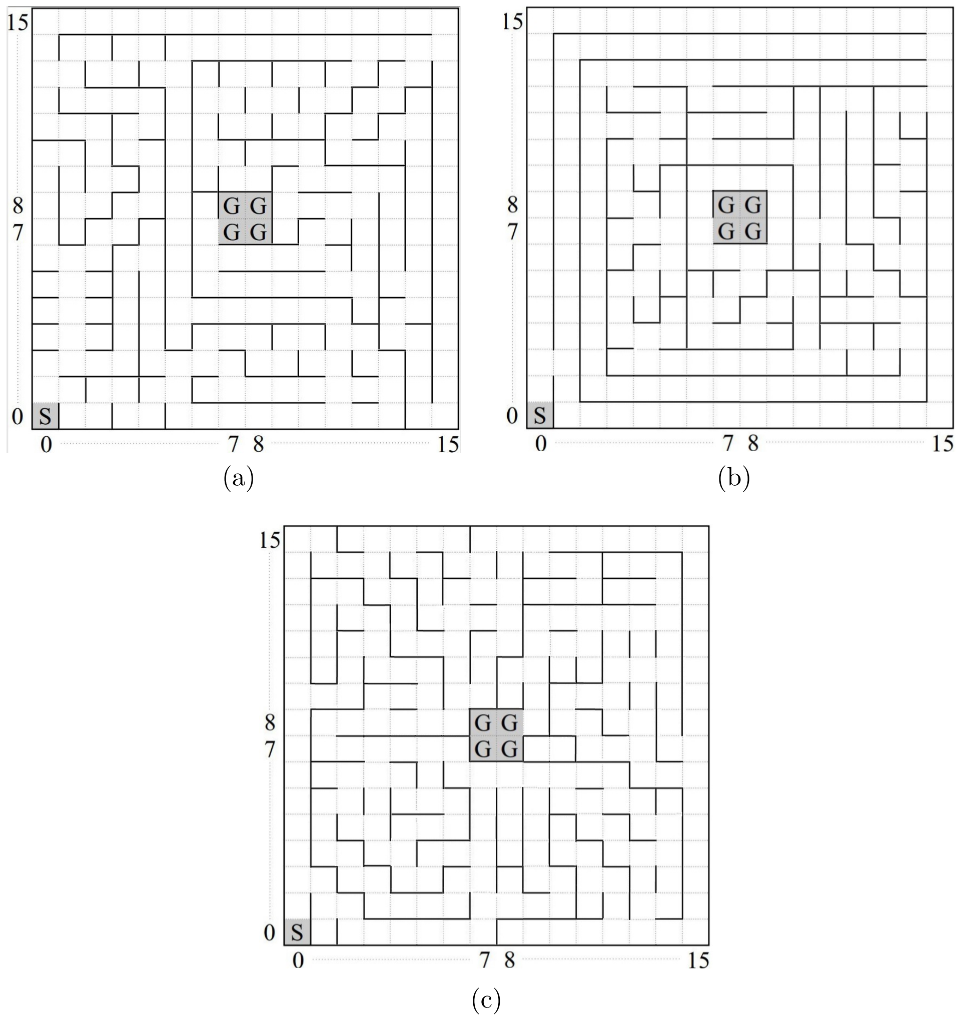


Figure 6.3: Maps of past micromouse competitions : (a) 2008 All Japan Micromouse Expert Contest (b) 2018 APEC Micromouse Contest. (c) 2018 UK Micromouse Finals Maze.

6.1.2.1 Wall Follower

The wall-follower algorithm presents a simple decision making, the robot moves through the unknown environment following the wall to its left. The complete description of its operation is presented in the section. When applying this algorithm in the labyrinths presented above, the robot traverses the trajectories presented in the Figure 6.4.

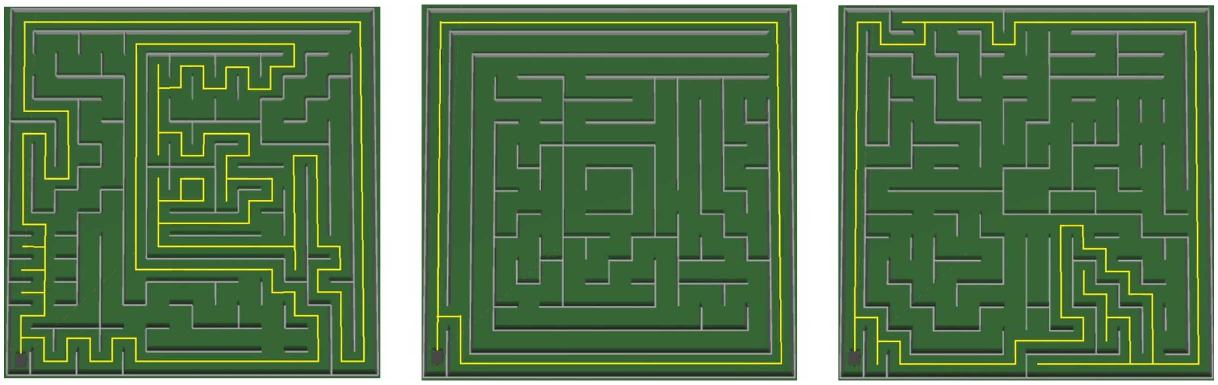


Figure 6.4: Path traversed using the wall-follower algorithm.

From these trajectories the maps presented in the Figure 6.5 were obtained. In these maps, the black cells contain obstacles, the white cells are free paths and the blue cells were not explored.

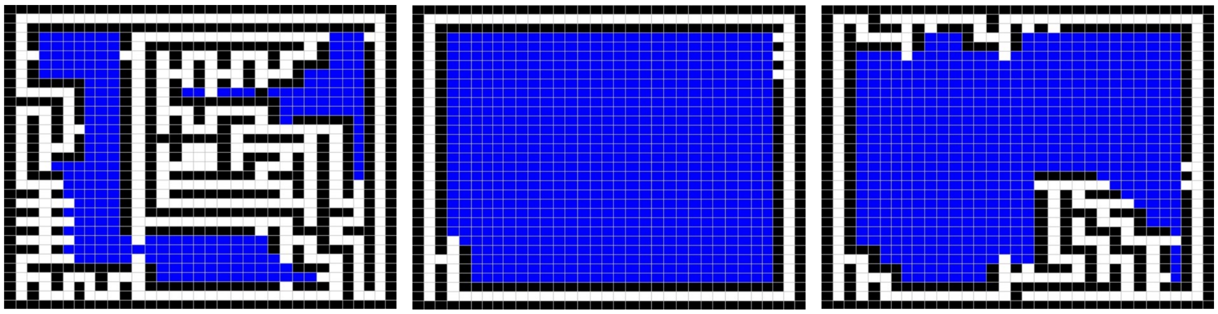


Figure 6.5: Maps generated by the wall follower algorithm

It is observed that after some time this algorithm goes into repetition, limiting the path traveled by the robot. This generates maps with little information due to the limited number of visited cells. Another disadvantage is the lack of assurance of finding a path between the starting point and the goal. Therefore, this algorithm proved to be insufficient to map the maze. Further information on the execution of this algorithm in the selected maps are found in the Table 6.1.

6.1.2.2 Wall Follower with repositioning algorithm

In order to prevent the robot from repeating the same paths during the mapping, a repositioning algorithm was added to the wall follower algorithm. This algorithm bases

Table 6.1: Results of the wall follower algorithm.

Maze	Mapped Percentage	Mapping Time (m)
2008 Japan	78.7	04:09
2018 APEC	33.9	00:49
2018 UK	47.4	02:21

its motion on the wall follower algorithm, however, when going through a path already covered a search algorithm is used to find the nearest unexplored entrance.The robot then traverses the generated path and restart the mapping of the environment from the entrance. When applying this algorithm in the same labyrinths, with the time limit of 7 minutes, the robot traverses the trajectories presented in the Figure 6.6.

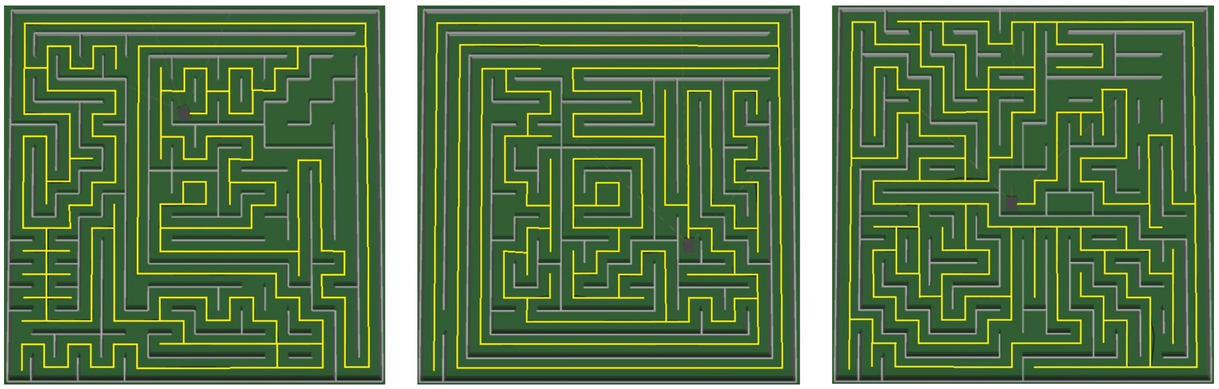


Figure 6.6: Path traversed using the wall-follower with repositioning algorithm.

From these trajectories the maps presented in the Figure 6.7 were obtained. In these maps, the black cells contain obstacles, the white cells are free paths, the blue cells were not explored and the green cells are unexplored entrances.

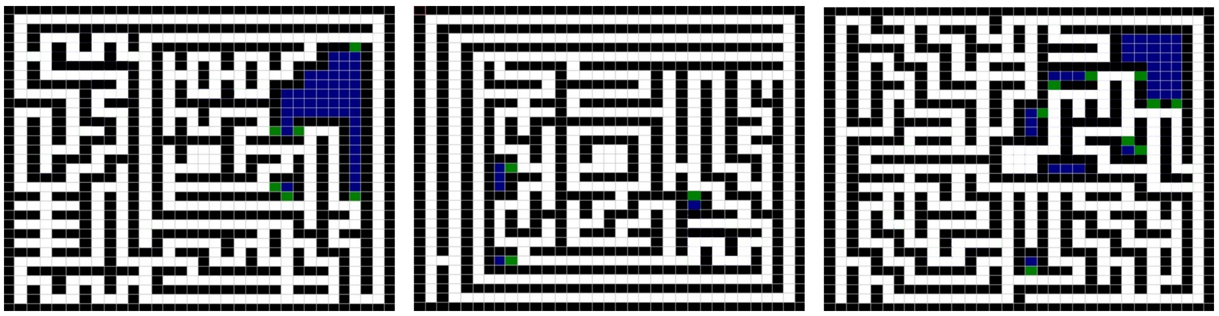


Figure 6.7: Maps generated by the wall follower with repositioning algorithm

This algorithm solves the repetition problem presented in the wall follower algorithm, increasing the available information about the labyrinth. Further information on the execution of this algorithm are found in the Table 6.2.

Table 6.2: Results of the wall follower with repositioning algorithm.

Maze	Mapped Percentage	Time to Fully Map the Maze (m)
2008 Japan	95.1	08:29
2018 APEC	99.2	07:52
2018 UK	95.68	08:43

6.1.3 Path Planning

In order to obtain the best path between the starting point and the goal a search algorithm is applied. However there are several algorithms that perform such a search. This section presents the results obtained by applying the four path planning algorithms described in the Sections 5.4 and 2.7. These algorithms are applied to the same maps used in the previous section, considering them fully mapped. The results are evaluated by the length of the route generated and the processing time required to obtain it.

6.1.3.1 2008 All Japan Micromouse Expert Contest

The results obtained in the application of the search algorithms are presented in the Figure 6.8, where red cells demonstrate the generated path, blues were visited in the search process and white cells were not visited.

Table 6.3 shows the number of cells visited, the number of cells to be traversed in the generated path and the processing time required to obtain the best path. The difference between the processing times of the modified A-Star with K equal to one and the regular A-Star is due to the different ordering processes, in the regular A-Star it is necessary to verify the cost of several cells while in the A-Star modified is only necessary to check some of them.

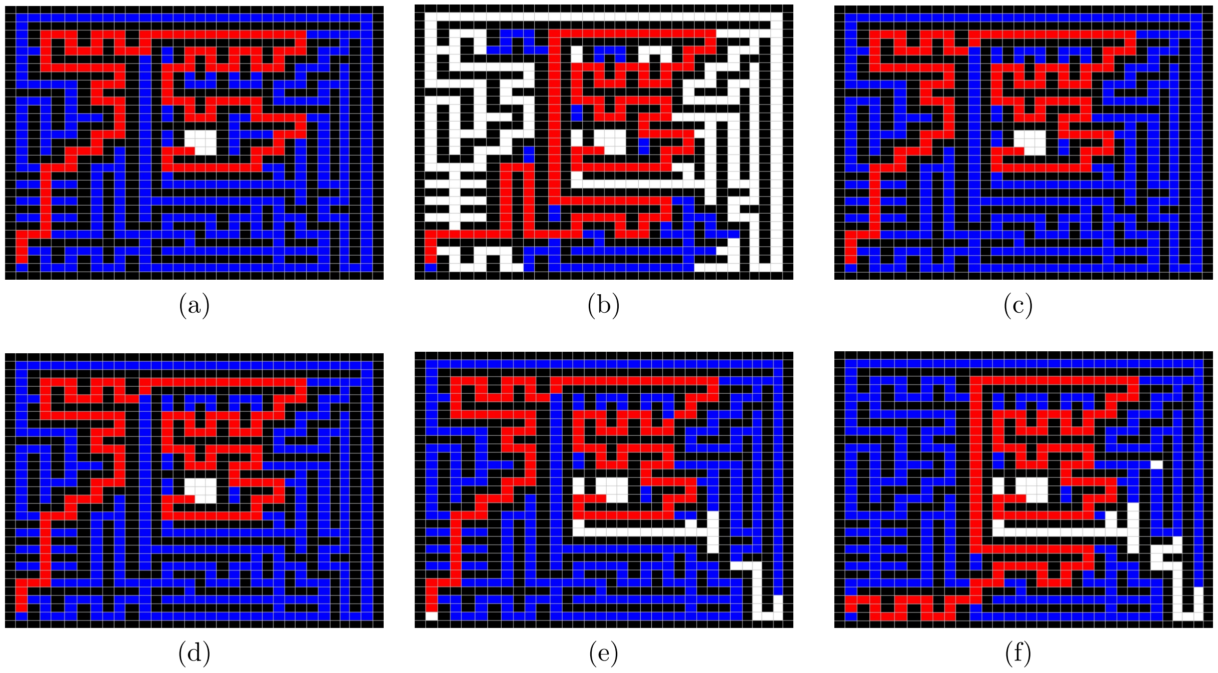


Figure 6.8: Search Algorithms applied on 2008 All Japan Micromouse: (a) Dijkstra's Algorithm (b) Best-first Search (c) A*. (d) A* Modified w/ K=1. (e) A* Modified w/ K=1.5. (f) A* Modified w/ K=2.

Table 6.3: Search Algorithms applied on 2008 All Japan Micromouse results

Search Algorithm	Processing Time	Visited Cells	Path Generated
Dijkstra's Algorithm	2043 ms	523	145
Best-first Search	657 ms	237	161
A*	2016 ms	523	145
A* Modified - K=1	1719 ms	523	145
A* Modified - K=1.5	1541 ms	489	145
A* Modified - K=2	1523 ms	479	153

6.1.3.2 2018 APEC Micromouse Contest

The results obtained in the application of the search algorithms are presented in the Figure 6.9, where red cells demonstrate the generated path, blues were visited in the search process and white cells were not visited.

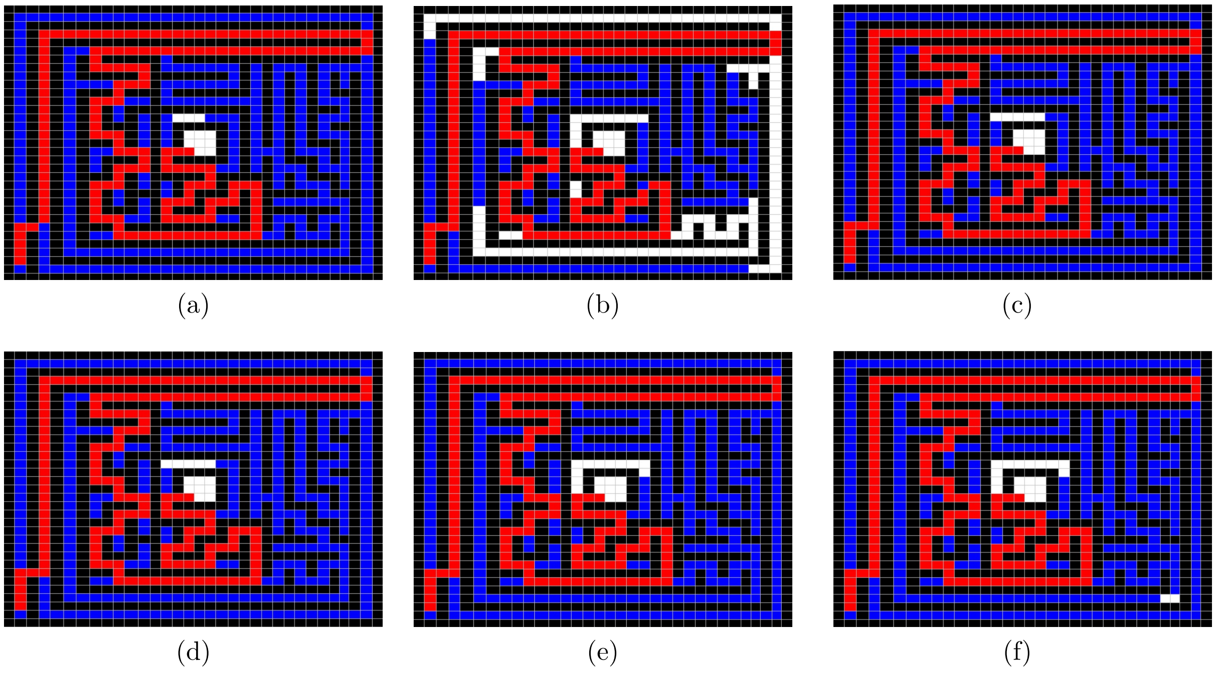


Figure 6.9: Search Algorithms applied on 2018 APEC Micromouse: (a) Dijkstra's Algorithm (b) Best-first Search (c) A*. (d) A* Modified w/ K=1. (e) A* Modified w/ K=1.5. (f) A* Modified w/ K=2.

Table 6.4 shows the number of cells visited, the number of cells to be traversed in the generated path and the processing time required to obtain the best path.

Table 6.4: Search Algorithms applied on 2018 APEC Micromouse results

Search Algorithm	Processing Time	Visited Cells	Path Generated
Dijkstra's Algorithm	1752 ms	512	173
Best-first Search	1200 ms	377	173
A*	1667 ms	509	173
A* Modified - K=1	1492 ms	509	173
A* Modified - K=1.5	1476 ms	503	173
A* Modified - K=2	1451 ms	500	173

6.1.3.3 2018 UK Micromouse

The results obtained in the application of the search algorithms are presented in the Figure 6.10, where red cells demonstrate the generated path, blues were visited in the search process and white cells were not visited.

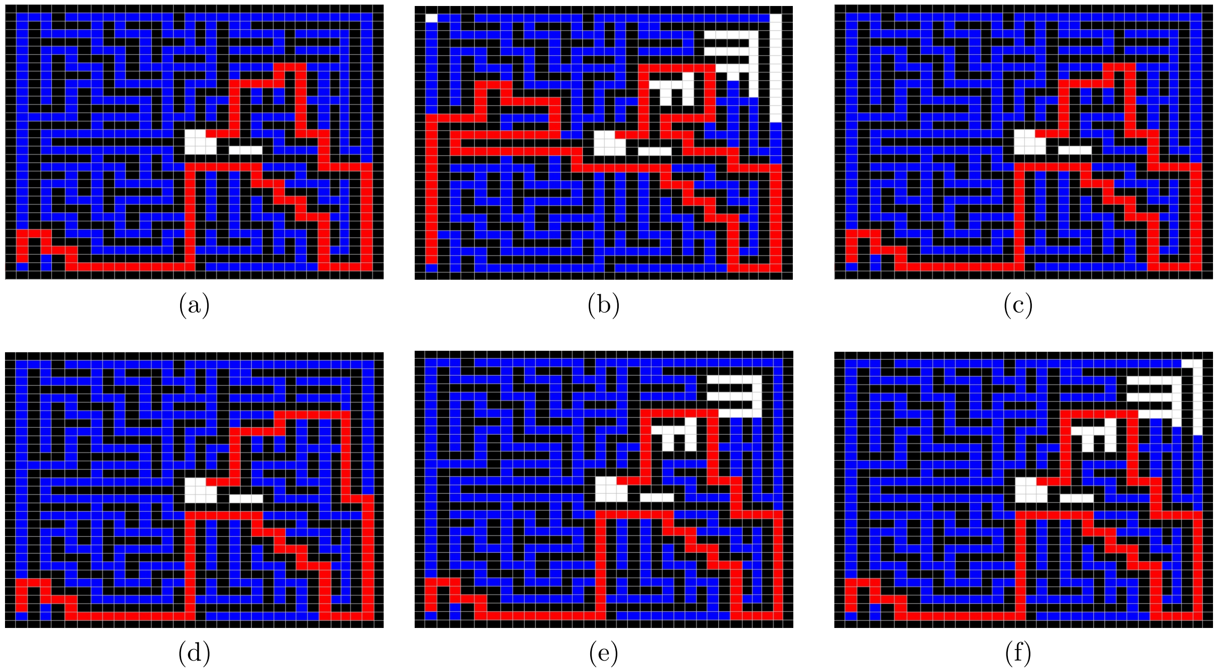


Figure 6.10: Search Algorithms applied on 2018 UK Micromouse : (a) Dijkstra's Algorithm (b) Best-first Search (c) A*. (d) A* Modified w/ K=1. (e) A* Modified w/ K=1.5. (f) A* Modified w/ K=2.

Table 6.5 shows the number of cells visited, the number of cells to be traversed in the generated path and the processing time required to obtain the best path.

Table 6.5: Search Algorithms applied on 2018 UK Micromouse results

Search Algorithm	Processing Time	Visited Cells	Path Generated
Dijkstra's Algorithm	2109 ms	528	109
Best-first Search	1831 ms	491	145
A*	2015 ms	528	109
A* Modified w/ K=1	1766 ms	528	109
A* Modified w/ K=1.5	1697 ms	503	109
A* Modified w/ K=2	1640 ms	491	109

6.1.3.4 Comparison

Among the algorithms evaluated, the modified A* and the best-first search obtained the shortest processing times. However, as the shortest path is desired, the best-first search was disregarded, due to generating sub-optimal paths. In the evaluation of the K values in the modified A*, a reduction of the processing time was observed as the increase of K, but consequently an increase in the length of the trajectory generated occurred. Therefore, from the evaluated algorithms it was chosen to use the modified A* with $K = 1$.

6.1.4 Evaluation Run

Based on the path obtained, the last run is performed. In this race of evaluation the time to reach the goal is verified and the competitor with the shortest time wins. The trajectories performed in the final runs of the mazes previously presented are shown in Figure 6.11.

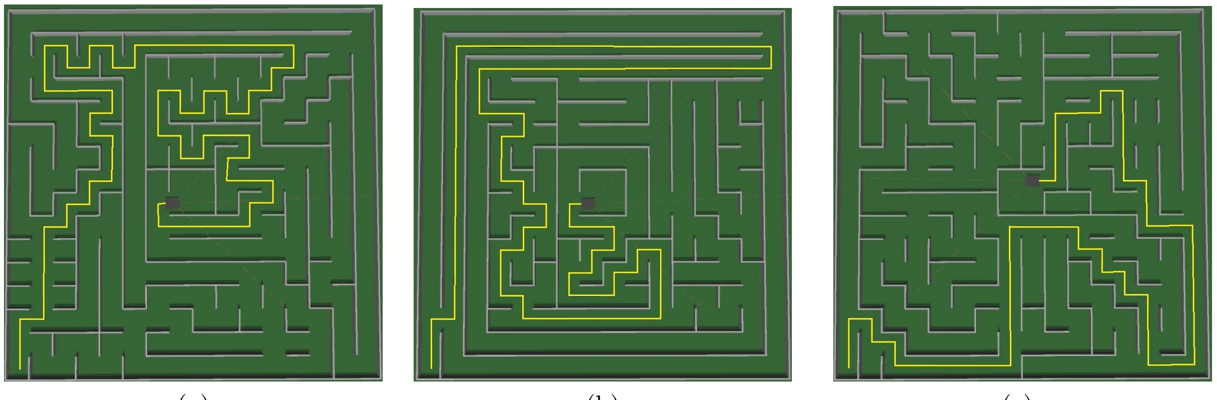

(a) (b) (c)

Figure 6.11: Trajectories performed on the evaluation run in the mazes : (a) 2008 All Japan Micromouse Expert Contest (b) 2018 APEC Micromouse Contest. (c) 2018 UK Micromouse Finals Maze.

The distance traveled and the time spend to reach the center of each labyrinth are shown in the Table 6.6.

Table 6.6: Evaluation run results.

Maze	Time to complete the run	Distance performed
2008 All Japan Micromouse	82.20 s	12.87 m
2018 APEC Micromouse Contest	75.33 s	15.39 m
2018 UK Micromouse	49.94 s	9.63 m

6.2 Real Robot

To perform the tests in the real robot, a maze with reduced dimensions was assembled as can be seen in Figure 6.12. This maze was built on a wooden surface. The walls were made of Medium-density fibreboard (MDF) having 5 cm of height and 3 mm of thickness. The labyrinth is formed by 5 × 5 square cells of 174 × 174 cm, the dimensions of the cells were reduced to maintain the internal value of the cell in 168 × 168 cm [7].

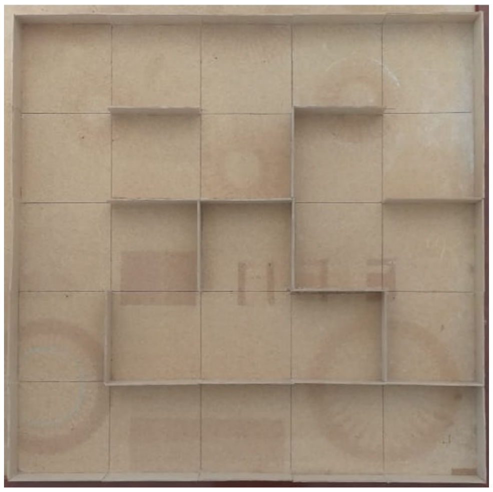


Figure 6.12: Assembled maze.

In this maze, the robot was able to perform the complete mapping in 48.2 s, the generated map is shown in the Figure 6.13a. Based on this map, the modified A* algorithm was applied obtaining the trajectory shown in Figure 6.13b. The evaluation run was performed (Figure 6.14) with the best time of 19.6 s.

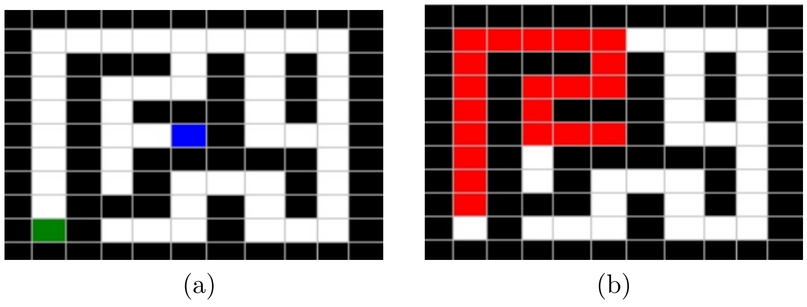


Figure 6.13: Real maze: (a) Map. (b) Path planning.

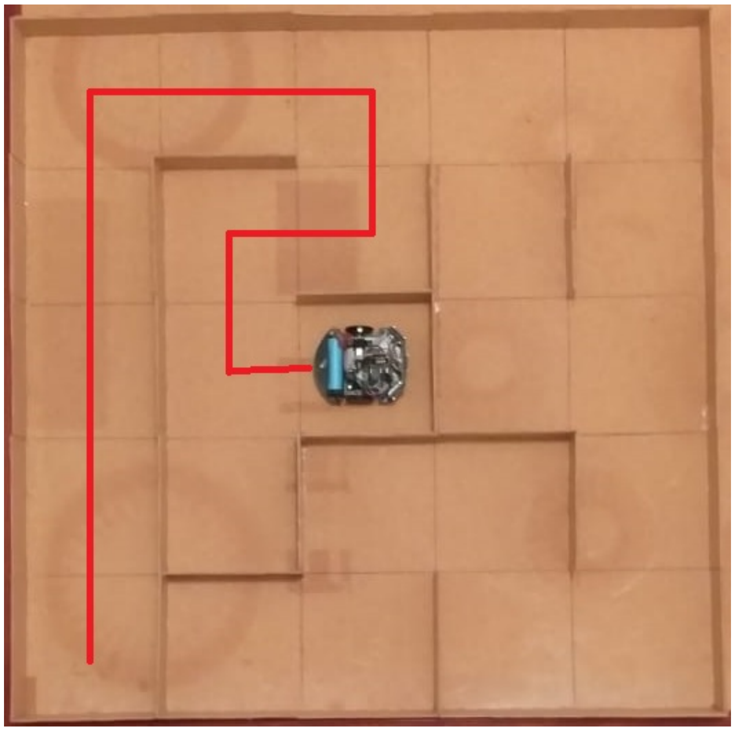


Figure 6.14: Trajectory performed in the evaluation run.

Chapter 7

Conclusions and Future Work

7.1 Developed Work

A methodology has been developed that allows a small robot to self-locate, map, find the best path between the starting point and the center of a maze, and cross it without colliding. Areas of mobile robotics such as simultaneous location and mapping, path planning and control were employed with the purpose of fulfill the required tasks.

Two different motion methodologies in an unknown environment were compared in order to obtain as much information as possible from the labyrinth during the mapping period. Among the algorithms tested, the replanning algorithm demonstrated a high capacity of collecting information about the maze and in all verified cases was able to locate the objective in less than 7 minutes.

An optimization for the A * search algorithm has been developed and compared with other well-known search algorithms, seeking to generate the shortest path with the lowest processing time possible. This optimization proved to maintain the path quality with the lowest processing time among these algorithms.

Finally, the real and simulated robot were tested in the evaluation run and were able to carry out the planned route.

7.2 Future Works

The present work has the potential to continue to be developed, especially optimizing the time of the evaluation run. Among the optimizations to be performed are:

- Non-stopping 90 degrees turns;
- Implementation of 45-degree curves;
- Addition of a new cost to the search algorithm to reduce the number of curves, since the curves require more time to execute;
- Insert a fan in the real robot, allowing an increase in robot speed without affecting its location.

Bibliography

[1] UTAD, *Utad micromouse portuguese contest*, "Accessed on 12/06/2018 at 09:53 p.m.". [Online]. Available: `https://www.micromouse.utad.pt/`.

[2] F. Tobe, *Robotics industry growing faster than expected*, "Accessed on 06/12/2018 at 10:25 a.m.", 2017. [Online]. Available: `https://www.therobotreport.com/robotics-industry-growing-faster-than-expected/`.

[3] S. G. Kibler, A. E. Hauer, D. S. Giessel, C. S. Malveaux, and D. Raskovic, "Ieee micromouse for mechatronics research and education," in *Mechatronics (ICM), 2011 IEEE International Conference on*, IEEE, 2011, pp. 887–892.

[4] S. Mishra and P. Bande, "Maze solving algorithms for micro mouse," in *Signal Image Technology and Internet Based Systems, 2008. SITIS'08. IEEE International Conference on*, IEEE, 2008, pp. 86–93.

[5] A. M. CONTEST, *Apec 31th annual micromouse contest*, 2017. [Online]. Available: `https://www.apec-conf.org/Portals/0/APEC%202019/APEC%202017%20Micromouse%20Results.pdf`.

[6] S. Yadav, K. K. Verma, and S. Mahanta, "The maze problem solved by micro mouse," *International Journal of Engineering and Advanced Technology (IJEAT) ISSN*, pp. 2249–8958, 2012.

[7] M. P. Contest, *Rules of the micromouse portuguese competest*, "Accessed on 05/09/2018 at 02:30 p.m.". [Online]. Available: `http://www.micromouse.utad.pt/?page_id=504&lang=en`.

[8] S. Behnke, “Robot competitions-ideal benchmarks for robotics research,” in *Proc. of IROS-2006 Workshop on Benchmarks in Robotics Research*, Institute of Electrical and Electronics Engineers (IEEE), 2006.

[9] L. B. Almeida, J. Azevedo, C. Cardeira, P. Costa, P. Fonseca, P. Lima, A. F. Ribeiro, and V. Santos, “Mobile robot competitions: Fostering advances in research, development and education in robotics,” 2000.

[10] M. Kandlhofer and G. Steinbauer, “Evaluating the impact of educational robotics on pupils’ technical-and social-skills and science related attitudes,” *Robotics and Autonomous Systems*, vol. 75, pp. 679–685, 2016.

[11] J. Gonçalves, J. Lima, P. J. Costa, and A. P. Moreira, “Modeling and simulation of the emg30 geared motor with encoder resorting to simtwo: The official robot@ factory simulator,” in *Advances in Sustainable and Competitive Manufacturing Systems*, Springer, 2013, pp. 307–314.

[12] R. Federation, *Robocup humanoid league*, `https://www.robocuphumanoid.org/`, 2015.

[13] K. Osuka, R. Murphy, and A. C. Schultz, “Usar competitions for physically situated robots,” *IEEE Robotics & Automation Magazine*, vol. 9, no. 3, pp. 26–33, 2002.

[14] R. Federation, *Robocuprescue*, `https://www.robocup.org/leagues/10`, 2016.

[15] ResCon, *19th rescue robot contest*, `https://www.rescue-robot-contest.org/19th-contest/about/`, 2018.

[16] P. J. Costa, N. Moreira, D. Campos, J. Gonçalves, J. Lima, and P. L. Costa, “Localization and navigation of an omnidirectional mobile robot: The robot@ factory case study,” *IEEE Revista Iberoamericana de Tecnologias del Aprendizaje*, vol. 11, no. 1, pp. 1–9, 2016.

[17] S. Thrun, M. Montemerlo, H. Dahlkamp, D. Stavens, A. Aron, J. Diebel, P. Fong, J. Gale, M. Halpenny, G. Hoffmann, *et al.*, “Stanley: The robot that won the darpa grand challenge,” *Journal of field Robotics*, vol. 23, no. 9, pp. 661–692, 2006.

[18] S. J. Russell and P. Norvig, *Artificial intelligence: a modern approach*. Malaysia; Pearson Education Limited, 2016.

[19] L. Piardi, "Application of a mobile robot to spatial mapping of radioactive substances in indoor environment," PhD thesis, 2018.

[20] R. Siegwart, I. R. Nourbakhsh, D. Scaramuzza, and R. C. Arkin, *Introduction to autonomous mobile robots*. MIT press, 2011.

[21] M. Hutson, *Curiosity rover decides—by itself—what to investigate on mars*, "Accessed on 05/06/2018 at 08:40 p.m.". [Online]. Available: `https://www.sciencemag.org/news/2017/06/curiosity-rover-decides-itself-what-investigate-mars`.

[22] L. Bruzzone and G. Quaglia, "Locomotion systems for ground mobile robots in unstructured environments," *Mechanical Sciences*, vol. 3, no. 2, pp. 49–62, 2012.

[23] R. R. Murphy, "Rescue robotics for homeland security," *Communications of the ACM*, vol. 47, no. 3, pp. 66–68, 2004.

[24] G. Quaglia, L. Bruzzone, G. Bozzini, R. Oderio, and R. P. Razzoli, "Epi. q-tg: Mobile robot for surveillance," *Industrial Robot: An International Journal*, vol. 38, no. 3, pp. 282–291, 2011.

[25] K. Nagatani, S. Kiribayashi, Y. Okada, K. Otake, K. Yoshida, S. Tadokoro, T. Nishimura, T. Yoshida, E. Koyanagi, M. Fukushima, *et al.*, "Emergency response to the nuclear accident at the fukushima daiichi nuclear power plants using mobile rescue robots," *Journal of Field Robotics*, vol. 30, no. 1, pp. 44–63, 2013.

[26] E. E. Cepolina and M. U. Hemapala, "Power tillers for demining: Blast test," *International Journal of Advanced Robotic Systems*, vol. 4, no. 2, p. 28, 2007.

[27] R. González, F. Rodrıguez, J. Sánchez-Hermosilla, and J. Donaire, "Navigation techniques for mobile robots in greenhouses," *Applied Engineering in Agriculture*, vol. 25, no. 2, pp. 153–165, 2009.

[28] P. Morin and C. Samson, "Motion control of wheeled mobile robots," in *Springer Handbook of Robotics*, Springer, 2008, pp. 799–826.

[29] S. Thrun, "Learning metric-topological maps for indoor mobile robot navigation," *Artificial Intelligence*, vol. 99, no. 1, pp. 21–71, 1998.

[30] D. Kortenkamp, R. P. Bonasso, and R. Murphy, *Artificial intelligence and mobile robots: case studies of successful robot systems*. MIT Press, 1998.

[31] D. Kortenkamp and T. Weymouth, "Topological mapping for mobile robots using a combination of sonar and vision sensing," in *AAAI*, vol. 94, 1994, pp. 979–984.

[32] D. Pierce and B. Kuipers, "Learning to explore and build maps," in *AAAI*, vol. 94, 1994, pp. 1264–1271.

[33] J. Giesbrecht, "Global path planning for unmanned ground vehicles," DEFENCE RESEARCH and DEVELOPMENT SUFFIELD (ALBERTA), Tech. Rep., 2004.

[34] H. Durrant-Whyte and T. Bailey, "Simultaneous localization and mapping: Part i," *IEEE robotics & automation magazine*, vol. 13, no. 2, pp. 99–110, 2006.

[35] M. Montemerlo, S. Thrun, D. Koller, B. Wegbreit, *et al.*, "Fastslam: A factored solution to the simultaneous localization and mapping problem," *Aaai/iaai*, vol. 593598, 2002.

[36] M. G. Dissanayake, P. Newman, S. Clark, H. F. Durrant-Whyte, and M. Csorba, "A solution to the simultaneous localization and map building (slam) problem," *IEEE Transactions on robotics and automation*, vol. 17, no. 3, pp. 229–241, 2001.

[37] T. S. Levitt, "Qualitative navigation for mobile robots," *Int. J. Artificial Intelligence*, vol. 44, pp. 305–360, 1990.

[38] B. Kuipers and Y.-T. Byun, "A robot exploration and mapping strategy based on a semantic hierarchy of spatial representations," *Robotics and autonomous systems*, vol. 8, no. 1-2, pp. 47–63, 1991.

[39] B. Yamauchi, A. Schultz, and W. Adams, “Mobile robot exploration and map-building with continuous localization,” in *Robotics and Automation, 1998. Proceedings. 1998 IEEE International Conference on*, IEEE, vol. 4, 1998, pp. 3715–3720.

[40] S. Garrido, L. Moreno, M. Abderrahim, and F. Martin, “Path planning for mobile robot navigation using voronoi diagram and fast marching,” in *Intelligent Robots and Systems, 2006 IEEE/RSJ International Conference on*, IEEE, 2006, pp. 2376–2381.

[41] C. K. Yap, “Algorithmic motion planning,” *Advances in robotics*, vol. 1, pp. 95–143, 1987.

[42] R. A. Brooks and T. Lozano-Perez, “A subdivision algorithm in configuration space for findpath with rotation,” *IEEE Transactions on Systems, Man, and Cybernetics*, no. 2, pp. 224–233, 1985.

[43] M. Mekni and P. Graniero, “A multiagent geosimulation approach for intelligent sensor web management,” *International Journal of Distributed Sensor Networks*, vol. 6, no. 1, p. 846 820, 2010.

[44] H. Noborio, T. Naniwa, and S. Arimoto, “A quadtree-based path-planning algorithm for a mobile robot,” *Journal of Robotic Systems*, vol. 7, no. 4, pp. 555–574, 1990.

[45] F. Avnaim, J.-D. Boissonnat, and B. Faverjon, “A practical exact motion planning algorithm for polygonal objects amidst polygonal obstacles,” in *Robotics and Automation, 1988. Proceedings., 1988 IEEE International Conference on*, IEEE, 1988, pp. 1656–1661.

[46] H. Miao and Y.-C. Tian, “Robot path planning in dynamic environments using a simulated annealing based approach,” 2008.

[47] S. H. Tang, W. Khaksar, N. Ismail, and M. Ariffin, “A review on robot motion planning approaches,” *Pertanika Journal of Science and Technology*, vol. 20, no. 1, pp. 15–29, 2012.

[48] H. M. Choset, S. Hutchinson, K. M. Lynch, G. Kantor, W. Burgard, L. E. Kavraki, and S. Thrun, *Principles of robot motion: theory, algorithms, and implementation.* MIT press, 2005.

[49] V. Boor, M. H. Overmars, and A. F. Van Der Stappen, "The gaussian sampling strategy for probabilistic roadmap planners," in *Robotics and automation, 1999. proceedings. 1999 ieee international conference on*, IEEE, vol. 2, 1999, pp. 1018–1023.

[50] O. Khatib, "Real-time obstacle avoidance for manipulators and mobile robots," in *Robotics and Automation. Proceedings. 1985 IEEE International Conference on*, IEEE, vol. 2, 1985, pp. 500–505.

[51] R. Volpe, "Real and artificial forces in the control of manipulators: Theory and experiments," PhD thesis, PhD thesis, Carnegie Mellon University, Department of Physics, 1990.

[52] J.-C. Latombe, *Robot motion planning.* Springer Science & Business Media, 2012, vol. 124.

[53] Q. Zhu, Y. Yan, and Z. Xing, "Robot path planning based on artificial potential field approach with simulated annealing," in *Intelligent Systems Design and Applications, 2006. ISDA'06. Sixth International Conference on*, IEEE, vol. 2, 2006, pp. 622–627.

[54] M. G. Park and M. C. Lee, "Experimental evaluation of robot path planning by artificial potential field approach with simulated annealing," in *SICE 2002. Proceedings of the 41st SICE Annual Conference*, IEEE, vol. 4, 2002, pp. 2190–2195.

[55] A. Gasparetto, P. Boscariol, A. Lanzutti, and R. Vidoni, "Path planning and trajectory planning algorithms: A general overview," in *Motion and operation planning of robotic systems*, Springer, 2015, pp. 3–27.

[56] S.-G. Cui, H. Wang, and L. Yang, "A simulation study of a-star algorithm for robot path planning," in *16th international conference on mechatronics technology, PP*, 2012, pp. 506–510.

[57] A. P. Moreira, P. Costa, and P. Costa, “Real-time path planning using a modified a* algorithm,” in *Proceedings of ROBOTICA 2009-9th Conference on Mobile Robots and Competitions*, 2009.

[58] R. J. Szczerba, P. Galkowski, I. S. Glicktein, and N. Ternullo, “Robust algorithm for real-time route planning,” *IEEE Transactions on Aerospace and Electronic Systems*, vol. 36, no. 3, pp. 869–878, 2000.

[59] J. Pearl, “Heuristics: Intelligent search strategies for computer problem solving,” 1984.

[60] WEMOS, *Wemos d1 mini*, "Accessed on 25/06/2018 at 07:30 p.m.". [Online]. Available: `https://wiki.wemos.cc/products:d1:d1_mini`.

[61] ——, *Motor shield*, "Accessed on 25/06/2018 at 07:50 p.m.". [Online]. Available: `https://wiki.wemos.cc/products:d1_mini_shields:motor_shield`.

[62] ——, *Battery shield*, "Accessed on 25/06/2018 at 08:00 p.m.". [Online]. Available: `https://wiki.wemos.cc/products:d1_mini_shields:battery_shield`.

[63] M. K. Kazimierczuk, *Pulse-width modulated DC-DC power converters*. John Wiley & Sons, 2015.

[64] R. Murphy, R. R. Murphy, and R. C. Arkin, *Introduction to AI robotics*. MIT press, 2000.

[65] A. Y. Hata, “Mapeamento de ambientes externos utilizando robôs móveis,” PhD thesis, Universidade de São Paulo, 2010.

[66] J. Borenstein, H. R. Everett, L. Feng, and D. Wehe, “Mobile robot positioning: Sensors and techniques,” *Journal of robotic systems*, vol. 14, no. 4, pp. 231–249, 1997.

[67] C. Gourley and M. Trivedi, “Sensor based obstacle avoidance and mapping for fast mobile robots,” in *Robotics and Automation, 1994. Proceedings., 1994 IEEE International Conference on*, IEEE, 1994, pp. 1306–1311.

[68] J. Borenstein and L. Feng, “Measurement and correction of systematic odometry errors in mobile robots,” *IEEE Transactions on robotics and automation*, vol. 12, no. 6, pp. 869–880, 1996.

[69] J. Lima, J. Gonçalves, P. J. Costa, and A. P. Moreira, “Modeling and simulation of a laser scanner sensor: An industrial application case study,” in *Advances in Sustainable and Competitive Manufacturing Systems*, Springer, 2013, pp. 245–258.

[70] L. Piardi, J. Lima, and P. Costa, “Development of a ground truth localization system for wheeled mobile robots in indoor environments based on laser range-finder for low-cost systems,” *International Conference on Informatics in Control, Automation and Robotics*, 2018.

[71] T. Pinho, A. P. Moreira, and J. Boaventura-Cunha, “Framework using ros and simtwo simulator for realistic test of mobile robot controllers,” in *CONTROLO’2014– Proceedings of the 11th Portuguese Conference on Automatic Control*, Springer, 2015, pp. 751–759.

[72] C. Paulo, G. José, L. José, and M. Paulo, “Simtwo realistic simulator: A tool for the development and validation of robot software,” *Theory and Applications of Mathematics & Computer Science*, vol. 1, no. 1, pp. 17–33, 2011.

[73] micromouseonline, *Github repository with maze files to micromouse*, `https://github.com/micromouseonline/mazefiles`, 2018.

[74] J. Lima, “Construção de um modelo realista e controlo de um robô humanóide,” 2009.

[75] P. Sarhadi and S. Yousefpour, “State of the art: Hardware in the loop modeling and simulation with its applications in design, development and implementation of system and control software,” *International Journal of Dynamics and Control*, vol. 3, no. 4, pp. 470–479, 2015.

[76] J. L. Burbank, W. Kasch, and J. Ward, *An introduction to network modeling and simulation for the practicing engineer*. John Wiley & Sons, 2011, vol. 5.

[77] M. Karpenko and N. Sepehri, "Hardware-in-the-loop simulator for research on fault tolerant control of electrohydraulic flight control systems," in *American Control Conference, 2006*, IEEE, 2006, 7–pp.

[78] M. Lee, H. Lee, K. S. Lee, S. Ha, J. Bae, J. Park, H. Park, H. Choi, and H. Chun, "Development of a hardware in the loop simulation system for electric power steering in vehicles," *International journal of Automotive technology*, vol. 12, no. 5, p. 733, 2011.

[79] G. Naus, J. Ploeg, M. Van de Molengraft, W. Heemels, and M. Steinbuch, "Design and implementation of parameterized adaptive cruise control: An explicit model predictive control approach," *Control Engineering Practice*, vol. 18, no. 8, pp. 882–892, 2010.

[80] M. O. Faruque and V. Dinavahi, "Hardware-in-the-loop simulation of power electronic systems using adaptive discretization," *IEEE transactions on industrial electronics*, vol. 57, no. 4, pp. 1146–1158, 2010.

[81] G. D. White, R. M. Bhatt, C. P. Tang, and V. N. Krovi, "Experimental evaluation of dynamic redundancy resolution in a nonholonomic wheeled mobile manipulator," *IEEE/ASME Transactions on Mechatronics*, vol. 14, no. 3, pp. 349–357, 2009.

[82] A. Martin and M. R. Emami, "Dynamic load emulation in hardware-in-the-loop simulation of robot manipulators," *IEEE Transactions on Industrial Electronics*, vol. 58, no. 7, pp. 2980–2987, 2011.

[83] M. tools, *Github repository with developed tools for the micromouse competition*, `https://github.com/P33a/SimTwo/releases/tag/2019Jan`, 2018.

[84] M. Rahman, "Autonomous maze solving robot," PhD thesis, May 2017. DOI: `10.13140/RG.2.2.34525.82403`.

[85] M. R. Nepali, N. Yadav, D. A. H. Prasad, and S. Balasubramaniam, "A novel wall following algorithm for mobile robots," *International Journal of Robotics and Automation (IJRA)*, vol. 5, no. 2, p. 15, 2014.

Appendix A

Publications

Eckert, L., Lima, J., Costa, A., Nakano, A. "Development of an Autonomous Mobile Robot with Planning and Location in a Structured Environment". In 1ª Escola de Verão e Simpósio de Dupla Diplomação 2018 (DD 2018).

Eckert, L., Piardi, L., Lima, J., Costa, P., Nakano, A. "An optimization approach on the robot maze path planning". In Encontro Nacional da Sociedade Portuguesa de Matemática (ENSPM 2018).

Lima, J., Costa, P., Costa, P., Eckert, L., Piardi, L., Moreira, A. P., Nakano, A. (2018, September). "A* Search Algorithm Optimization Path Planning in MobileRobots Scenarios". In International Conference of Numerical Analysis and Applied Mathematics (ICNAAM). Rhodes, Greece.

A.0.1 Paper Submitted and Accepted

Eckert, L., Piardi, L., Lima, J., Costa, P., Valente, A., Nakano, A. "3D Simulator Based on SimTwo to EvaluateAlgorithms in Micromouse Competition". In 7Th World Conference on Information Systems and Technologies (WCIST).

A.0.2 Papers Submitted and Awaiting Evaluation

Eckert, L., Piardi, L., Lima, J., Costa, P., Valente, A., Nakano, A. "3D Simulator with Hardware-in-the-Loop capability for the Micromouse Competition" in 18th IEEE International Conference on Autonomous Robot Systems and Competitions (ICARSC).