Using a mobile robot for hazardous substances detection in a factory environment

João Afonso Braun Neto

Dissertação apresentada à Escola Superior de Tecnologia e de Gestão do Instituto Politécnico de Bragança para obtenção do Grau de Mestre em Engenharia Industrial - Ramo de Engenharia Eletrotécnica.

Trabalho orientado por:

Professor PhD José Luis Sousa Magalhães Lima

Professor PhD José Cerqueira Gomes da Costa

Professor PhD Alberto Yoshihiro Nakano

Bragança

2019

Using a mobile robot for hazardous substances detection in a factory environment

João Afonso Braun Neto

Dissertação apresentada à Escola Superior de Tecnologia e de Gestão do Instituto Politécnico de Bragança para obtenção do Grau de Mestre em Engenharia Industrial - Ramo de Engenharia Eletrotécnica.

Trabalho orientado por:
Professor PhD José Luis Sousa Magalhães Lima
Professor PhD José Cerqueira Gomes da Costa
Professor PhD Alberto Yoshihiro Nakano

Bragança
2019

In the middle of difficulty lies opportunity. Albert Einstein.

Dedication

To my beloved mother who encouraged me through all this process and made possible through aiding me emotionally and financially. To my father who provided me with financially support and counseling. I dedicate this work to my grandparents as well who always encourage me to study and give the best of myself...

Acknowledgement

Throughout the stage of this dissertation research and writing I have received a great deal of support and assistance. I would first like to express my gratitude to my supervisor, PhD. José Lima at IPB in Portugal, whose expertise was invaluable in the formulating of the research topic and methodology in particular, providing technical council and help whenever I needed. I would like to thank my co-supervisor, PhD. Alberto Nakano at UTFPR in Brazil, for providing his knowledge in academic papers helping me correct my work and advising me. I would like to acknowledge PhD. Ana Pereira, at IPB in Portugal, who helped me with the gas search algorithm. Last but not least, my co-supervisor, PhD. Paulo Costa at FEUP in Portugal, who aided me with some challenges that I had throughout the work. I would like to acknowledge my colleagues at CeDRI who were always willing to help me and provided me with humorous conversations in the work intervals. You sure made this process lighter and amusing. In addition, I would like to thank my parents for their wise counsel and sympathetic ear. You are always there for me. Moreover, to my friends who were of great support in deliberating over our problems and findings, as well as providing with funny situations making the process easier. Finally, there is my girlfriend who was always there for me, encouraged me to pursue my dream and provided with advice and happy distractions to rest my mind outside of my research.

Abstract

Industries that work with toxic materials need extensive security protocols to avoid accidents. Instead of having fixed sensors, the concept of assembling the sensors on a mobile robot that performs the scanning through a defined path is cheaper, configurable and adaptable. This work describes a mobile robot, equipped with several gas sensors and LIDAR, that follows a trajectory based on waypoints, simulating a working Autonomous Guided Vehicle (AGV). At the same time, the robot keeps measuring for toxic gases. In other words, the robot follows the trajectory while the gas concentration is under a defined value. Otherwise, it starts the autonomous leakage search based on a search algorithm that allows to find the leakage position avoiding obstacles in real time. The proposed methodology is verified in simulation based on a model of the real robot. Therefore, three path plannings were developed and their performance compared. A Light Detection And Ranging (LIDAR) device was integrated with the path planning to propose an obstacle avoidance system with a dilation technique to enlarge the obstacles, thus, considering the robot's dimensions. Moreover, if needed, the robot can be remotely operated with visual feedback. In addition, a controller was made for the robot. Gas sensors were embedded in the robot with Finite Impulse Response (FIR) filter to process the data. A low cost AGV was developed to compete in Festival Nacional de Robótica (Portuguese Robotics Open) 2019 - Gondomar, describing the robot's control and software solution to the competition.

Keywords: Autonomous Mobile Robot, Obstacle Avoidance, Mobile Gas Search, Path planning, A*, RRT*, Light Detection and Ranging, Laser Ranger Finder, Automatic Guided Vehicle, Robotic Competition.

Resumo

As indústrias que trabalham com materiais tóxicos necessitam de extensos protocolos de segurança para evitar acidentes. Ao invés de ter sensores estáticos, o conceito de instalar sensores em um robô móvel que inspeciona através de um caminho definido é mais barato, configurável e adaptável. O presente trabalho descreve um robô móvel, equipado com vários sensores de gás e LIDAR, que percorre uma trajetória baseada em pontos de controle, simulando um AGV em trabalho. Em simultâneo são efetuadas medidas de gases tóxicos. Em outras palavras, o robô segue uma trajetória enquanto a concentração de gás está abaixo de um valor definido. Caso contrário, inicia uma busca autônoma de vazamento de gás com um algoritmo de busca que permite achar a posição do gás evitando os obstáculos em tempo real. A metodologia proposta é verificada em simulação. Três algoritmos de planejamento de caminho foram desenvolvidos e suas performances comparadas. Um LIDAR foi integrado com o planejamento de caminho para propôr um sistema de evitar obstáculos. Além disso, o robô pode ser operado remotamente com auxílio visual. Foi feito um controlador para o robô. Sensores de gás foram embarcados no robô com um filtro de resposta ao impulso finita para processar as informações. Um veículo guiado automático de baixo custo foi desenvolvido para competir no Festival Nacional de Robótica 2019 - Gondomar. O controle do veículo foi descrito com o programa de solução para a competição.

Keywords: Robô Autônomo Móvel, Evitar Obstáculos, Procura por Gás Móvel, Planejamento de Caminho, A*, RRT*, Light Detection and Ranging, Laser Ranger Finder, Veículo Guiado Automático, Competição de Robótica.

Contents

List of Tables

List of Figures

Acronyms

A* A Star.

ADC Analog to Digital Converter.

AGV Autonomous Guided Vehicle.

AGVs Autonomous Guided Vehicles.

AMR Autonomous Mobile Robot.

AMRs Autonomous Mobile Robots.

AUV Autonomous Underwater Vehicle.

CO Carbon Monoxide.

CO2 Carbon Dioxide.

DOF Degrees of Freedom.

FIR Finite Impulse Response.

FNR 2019 Festival Nacional de Robótica 2019.

FSM Finite State Machine.

GUI Graphical User Interface.

HiL Hardware in the Loop.

IC	Integrated Circuit.
INESC TEC	Institute for Systems and Computer Engineering, Technology and Science.
IPB	Instituto Politécnico de Bragança.
LIDAR	Light Detection And Ranging.
LMB	Left Mouse Button.
LPG	Liquefied Petroleum Gas.
LRF	Laser Ranger Finder.
NH3	Ammonia.
NOx	Nitrogen Oxides.
PRM	Probabilistic Roadmap.
RaF	Robot@Factory Lite.
RF	Radio Frequency.
RFID	Radio Frequency Identification.
RMB	Right Mouse Button.
ROS	Robot Operating System.
RRT	Rapidly Random Tree.
RRT*	Rapidly Random Tree Star.
RSS	Received Signal Strength.
RTLS	Real Time Locating System.

TDLAS Tunable Diode Laser Absorption Spectroscopy.

TDOA Time Difference of Arrivals.

ToF Time of Flight.

UAV Unmanned Aerial Vehicle.

UGV Unmanned Ground Vehicle.

UWB Ultra Wide Band.

VFH* Vector Field Histogram Star.

VFH+ Vector Field Histogram Plus.

WMR Wheeled Mobile Robot.

WMRs Wheeled Mobile Robots.

Chapter 1

Introduction

Robotics are an amusing area of study that is interdisciplinary. In specific, mobile robots are a reality nowadays and they are used in large scale from houses to industries, and even for scientific purposes. This work will cover important areas of study in the mobile robot subject: control, path planning, obstacle avoidance and autonomous search. The main objective of this work is to develop an approach for a mobile robot equipped with sensors following a working route, simulating an AGV in a factory, that autonomously searches avoiding obstacles, for a source of gas leakage when a certain threshold is passed. First, the approach is going to be tested in simulation and after that, real tests are going to be made.

1.1 Motivation

According to SMP Robotics [1], the control of air conditioning and toxic gas content at chemical and oil & gas plants is essential for protecting human health and the environment, and in certain cases, preventing accidents. Toxic gas content in the air is usually measured in close proximity to the equipment, and in general, gas inspection is performed by specialists with hand-held multi-gas monitors. This procedure can be automated by fitting a mobile robot with gas analyzers and sending it on patrol around the area [1].

Further, if the materials are highly toxic, the personnel could be evacuated and a specialist team must be called to remedy the situation. Therefore, autonomous mobile robots are developed to track toxic gases sources to aid the process of remediation and to avoid human fatalities. In addition, mobile robots could be remotely operated to find victims or to have a visiual feedback of the scenario. Thus, this work's idea is to embbed an AGV with these capabilities.

1.2 Objectives

Following the main objective stated in this Chapter, this work has several secondary objectives:

- Develop an application that serves as a human machine interface in real time;
- Develop a controller for the robot;
- Integrate, in the application, the option to the user select waypoints to generate a working route where the robot must pass;
- Implement path planning algorithms;
- Integrate LIDAR device with path planning algorithms to develop an obstacle avoidance system for the mobile robot;
- Adapt the existing real robot with gas sensors for real test;
- Develop an application that enables the user to remotely operate the robot with visual feedback;
- Develop a gradient gas search algorithm;
- Implement a FIR filter to process the gas sensors values.

1.3 Document Structure

This work has seven chapters and is structured as follows:

The introduction is presented in Chapter 1, which contains the work's motivation and objectives.

Chapter 2 describes the state of art regarding to Autonomous Guided Vehicles (AGVs) and Autonomous Mobile Robots (AMRs) areas of study, as both categories belong to this work scope. The theoretical background is presented, such as navigation, robot geometry, path planning and localization techniques and technologies with real applications. Moreover, the state of art in gas search using mobile robots is emphasized.

In addition, Chapter 3 displays the low cost AGV that was assembled to compete in Festival Nacional de Robótica 2019 (FNR 2019) in Gondomar, Portugal. Furthermore, the competition rules are addressed and the simulation platform as well as the robot's control and software solution for the competition are presented.

Chapter 4 presents the autonomous mobile robot system architecture with the hardware and its features such as the localization system and odometry.

Chapter 5, addresses the autonomous mobile robot navigation where the spline sampling, gas search algorithm, path planning algorithms and the robot's control are detailed.

Chapter 6, presents this work results.

Chapter 7, displays this work conclusions and possible future works.

Chapter 2

State of Art

As explained in Chapter 1, the mobile robot in this work is an AGV that will become autonomous in the present of a hazardous gas. In this regard, this chapter presents the related work of AGVs and AMRs and their respective technology such as, applications, path planning, localization methods, and obstacle avoidance. Moreover, the related work regarding hazardous substances detection will be described.

2.1 Applications of Autonomous Mobile Robots

According to Merriam-Webster dictionary, a robot is a machine that resembles a living creature in being capable of moving independently (as by walking or rolling on wheels) and performing complex actions (such as grasping and moving objects). However, for Wheeled Mobile Robots (WMRs), the definition can be stated as a vehicle which is capable of an autonomous motion. One of the most important tasks of an autonomous system of any kind is to acquire knowledge about its environment. This is done by taking measurements using various sensors and then extracting meaningful information from those measurements [2]. The first robots have emerged to operate in a cell isolated environment, most often within warehouses or industries [3].

Robot comes from the Czech word “robota” which means “forced work or labor”.

Therefore, most robots are designed to be applied in repetitive, dangerous jobs for humans and for tasks that does not fit a human. In this way, there are robots for exploring dangerous locations [4], measuring and localizing toxic and poisonous substances, performing dangerous tasks such as detecting bombs, destroying mines and surveillance [5], performing repetitive tasks in industries as building cars, monitoring places, among others. Furthermore, there are robots being applied to extreme precision tasks such as medical surgeries [6]. The developed robotic systems are usually implemented when, for the same work, the robot performs with a higher quality and speed or at a lower cost compared to human work [2].

The robot used in this work is a wheeled class and it was already been used in other works [3], [7], [8]. A Wheeled Mobile Robot (WMR) is applied to problems related to operations in complex environments such as hazardous and dangerous environments, unknown environments with dynamic obstacles, or planetary explorations to name a few examples [3]. The WMR category will be presented in Section 2.5, with its advantages for inspection, monitoring and surveillance accompanied by real application examples when possible.

The importance of inspection is acquired by helping to prevent incidents, injuries and illnesses. Through a critical examination of the workplace, inspections help to identify and report hazards for corrective action. Monitoring and surveillance act in a similar way, checking the environments in a recurring basis for factors that requires attention. These environments can be monitored for the presence of dust, people, humidity, temperature, fire risks, biological agents, radioactive substances or poisonous/toxic gases which is the case addressed in the present work.

Before the usage of mobile robots for inspection and surveillance, humans were the choice for these kind of tasks. Therefore, health hazards tasks or even life threatening tasks required extreme caution. For this reason, there is a need to avoid and replace the presence of humans in hazardous areas by robots. Solutions with WMR have been an ideal alternative for inspection and monitoring because they have a relatively low cost, avoid exposure of workers, are more accurate and perform in less time when compared to the classical approach [3].

WMRs can be autonomous as an Autonomous Mobile Robot (AMR), guided as an AGV or teleoperated as an Unmanned Aerial Vehicle (UAV), an Unmanned Ground Vehicle (UGV) or even an Autonomous Underwater Vehicle (AUV). The first two classified by being autonomous or guided and, in the other hand, the others being classified by the environment they travel. Take note, that the mobile robot can dispose of certain autonomy level and be remotely operated. An AGV is classified as robot that rely in navigation systems that guides them, such as: wires, magnetic strips and other forms. The most used navigation methods will be explained in Section 2.2. They have few applications and fixed routes. In addition, they are expensive, because as in industrial applications, the factory is constantly updated so that the robot can move, e.g, installing magnetic strips. In the other hand, AMR has intelligent navigation that works via maps that its software constructs on-site or via pre-loaded facility drawings. It uses data from cameras and built-in sensors, laser scanners as well as sophisticated software that enables it to detect its surroundings and chooses the most efficient route to the target. It works completely autonomously and if it happens to appear dynamic obstacles in its route, the AMR will safely maneuver around them, using the best alternative route with collision avoidance algorithms.

In this way, to WMRs operate, they must be able to acquire and use knowledge about the environment, estimate a position, control wheel speed, plan the route, have the ability to recognize obstacles and respond in real time to the different situations that may occur. All these features should work together [3].

2.1.1 S5 G Robot

The S5 G Robot seen in Figure 2.1 is capable to measure gas substances in industrial facilities and transmit the data remotely via WiFi communication. The robot is autonomous and can patrol the facility without the help of an operator. In addition, it has a collision avoidance system, and therefore, changes its route when obstacles are encountered.

Moreover, the robot stops periodically to perform gas measurements and can change its route depending on the wind direction [1]. When running out of batteries, it returns to its station to charge them. S5 G Gas Detector Robot is capable of performing video surveillance. In this reasoning, it is equipped with a panoramic video surveillance system fit for operation in dimly lit conditions [1]. Finally, the robot has a solar tracking system that ensures the optimal position and incident angle for the solar radiation. Therefore, the robot can operate without recharging its batteries for several weeks [1].

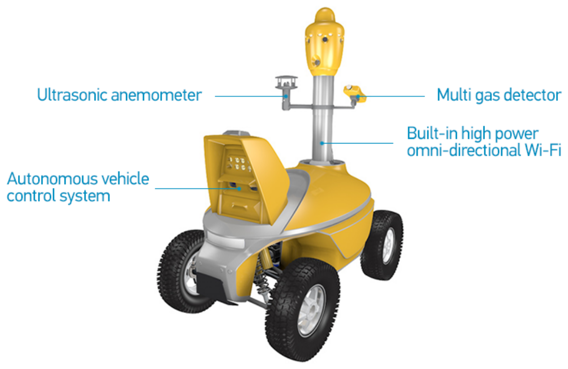

Ultrasonic anemometer
Multi gas detector
Built-in high power omni-directional Wi-Fi
Autonomous vehicle control system

Figure 2.1: S5 G Robot Illustration [1].

2.2 Application of Automonous Guided Vehicles

An AGV is an autonomous guided vehicle that follows a path. This equipment is widely used in industrial fields [9]. In these workplaces, they carry a wide variety of materials, even hazardous ones. Thus, since their introduction in 1953 [10], AGVs are now found in all types of industries, with the only restrictions due to the goods to be transported and spatial consideration [9].

An AGV is a vehicle that is controlled by a computer and is driven by a battery. Depending on their navigation technology, they can be restricted to follow determined paths or their routes can be changed infinitely.

According to H. Fazlollahtabar and M. Saidi-Mehrabad [9], the use of AGVs can be divided into four main areas:

1. supply and disposal at storage and production areas;
2. production-integrated application of AGV trucks as assembly platforms;
3. retrieval, especially in wholesale trade;
4. supply and disposal in special areas, such as hospitals and offices.

In this way, AGVs excel in applications that request repetitive movement of materials over distance, with a regular delivery schedule of loads in two shifts of work. Naturally, AGVs have been found to reduce the damage to inventory, make production scheduling more flexible, and reduce staffing needs [9].

The AGVs were traditionally employed in manufacturing systems until mid seventies [10]. After that, the advent of solid-state controls allowed systems to expand in capability and flexibility [10], thus, the result applications grew significantly. Nowadays, pharmaceutical, chemical, manufacturing, automotive, paper and print, food and beverage, warehouse and theme parks employ AGVs in their work. By default, one central server (central computer) administer AGVs in numerous complex tasks such as traffic, deadlock, obstacle avoidance and routing [9].

Navigation

The AGVs have several navigation methods that were introduced in the late 20th century and the early 21th century. A brief description of the most common, according to C. Russell [11] and D. Indelicato [12], will be as follows:

- Inductive Guidance;

  A wire is introduced into a section of the floor cut, forming the path the AGV must follow. This wire is used to transmit radio signals. In this way, a sensor embedded on the bottom of the AGV detects the relative position of the radio signal coming from the wire. With this information, the AGV steering control can follow the path. The sketch can be seen in Figure 2.2.

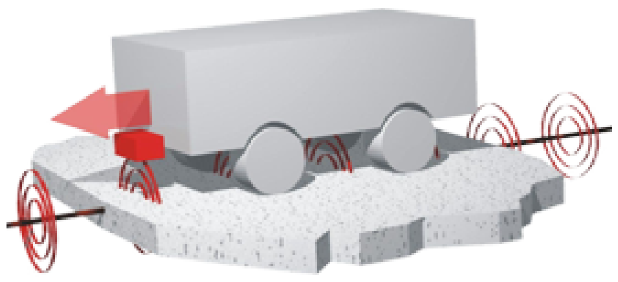


Figure 2.2: Inductive Guidance [13].

- Magnetic Tape Guidance;

  These AGVs as illustrated in Figure 2.3, use guide tape to follow their path. The tape can be magnetic or colored. Sensors appropriate to detect the type of tape are installed in the AGV. One advantage of the guide tape over the wired is that the tape can be easily reallocated, in the event that the route must be changed. Colored tape is cheaper than magnetic, however in areas with heavy traffic, the tape can wear off. In the other hand, as magnetic tape have dual polarity, it provides advantages as small pieces of tape can be put to change states of the AGV based on its polarity or sequence of “tags”. In addition, magnetic bars can be placed under the floor in the same way as the wired navigation method. The difference is that they are unpowered by electricity. The control for both guidance tapes is by computing its lateral distance in relation with the center of the tape, thus, providing to the steering control the information needed to drive the vehicle.

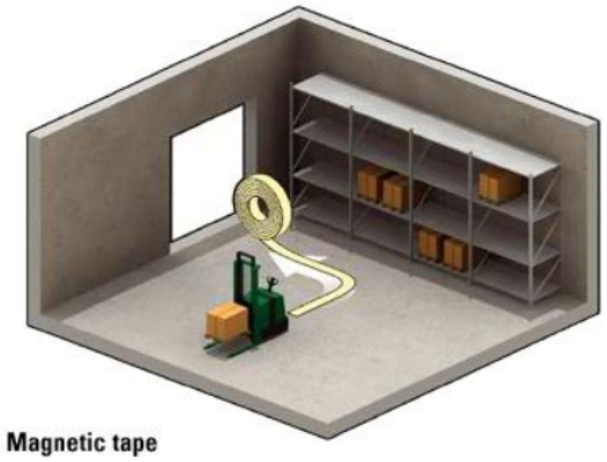


Figure 2.3: Magnetic Tape Guidance [14].

- Laser Guidance;

This method of navigation works by installing on the walls of the environment some poles and fixed bulkheads with reflectors. In this sense, the AGV is equipped with a laser transmitter and receiver in a rotating turret. The AGV can compute its location in the environment by using multilateration, thus, estimating its position in relation to the position of the reflectors and comparing with a map stored inside the AGV. Then, this information is compared with the route that the vehicle must follow. Soon after, the steering control acts to make the vehicle follow the path. The most common used laser transmitters are the modulated lasers and the pulsed lasers. The modulated lasers provide better accuracy than the pulsed ones. Figure 2.4 illustrates the technology.

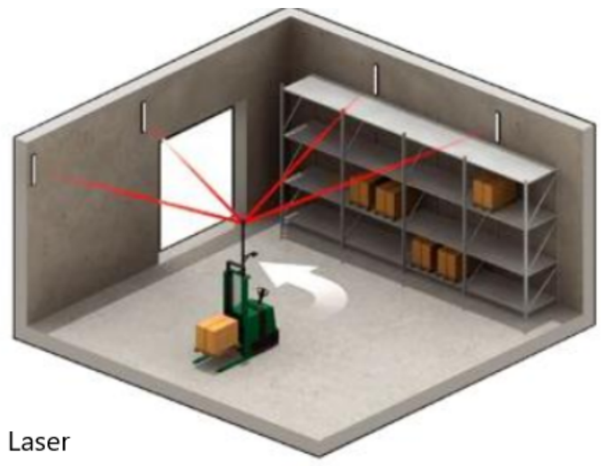


Figure 2.4: Laser Guidance [14].

- Inertial navigation;

  The inertial navigation guidance method works by using markers and a gyroscope so that the AGV can locate itself [15]. The inertial sensor can detect minimum change in the direction of the vehicle. This way, the AGV can correct its route. This method presents advantages to all environments, especially where magnetic, optic floor tape or laser targets are not feasible or desirable [15]. See Figure 2.5.

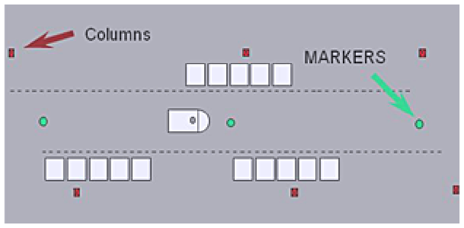


Figure 2.5: Inertial navigation [15].

- Natural Navigation;

  This method use range-finder sensors and gyroscopes with Monte-Carlo/Markov techniques to locate itself in the environment, planning the shortest path to the target. These systems are easier to install and can handle other AGVs failing on the track as they can plan a route around the failed AGV. In conclusion, they are highly flexible. Figure 2.6 shows the method.

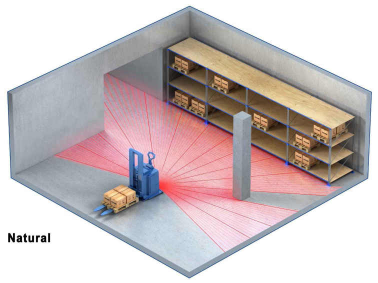


Figure 2.6: Natural Navigation [14].

- Vision guidance;

  The AGVs “replay” their route by comparing the features of the environment with previous camera recordings. These types of AGVs build a 3D map using 360° camera, allowing them to follow the route without any type of positioning system or landmark, that is, these systems can be employed without any change in the environment [16].

- Geoguidance.

  This method uses a laser navigation scanner to map the work environment before entering in action. After that, the map is then polished through software and virtual paths are added to the map. That enables the AGV to identify objects in the warehouse such as walls, racks and columns. Using these objects as fixed references, the vehicle can locate itself in the environment in real time and plan its route by comparing what the robot sees (laser scanner) and its reference map [17]. See Figure 2.7.

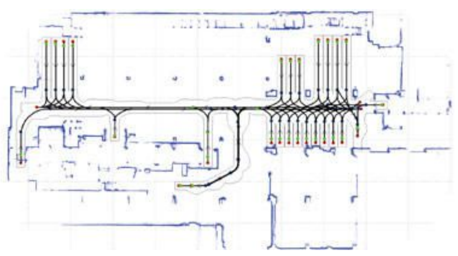


Figure 2.7: Geoguidance AGV reference map [17].

Steering systems

The AGVs can be equipped with several different steering systems. A brief explanation with some of the most common will be presented below.

The differential speed control is the most common [12]. This system consists of two independent driver wheels. The AGV can turn by accelerating the wheels at different speeds, thus, generating angular velocity. It works better in looped system and tugger AGVs, however, presents turning limitations in comparison to other steering systems [18]. The scheme can be seen in Figure 2.8 (a).

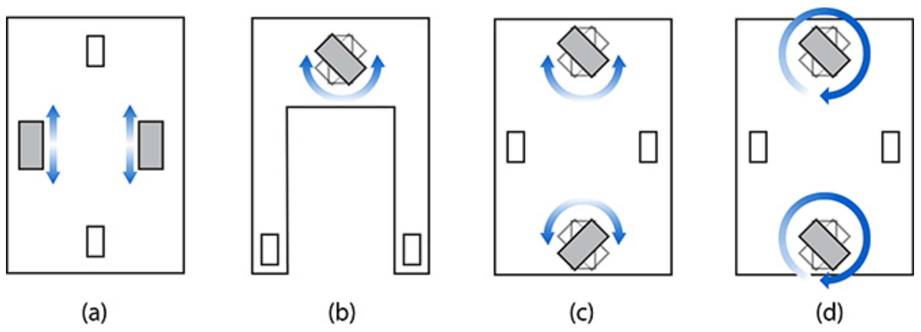


Figure 2.8: a) Differential steering. b) Single steered wheel with two fixed casters. c) Two steered wheels with 180° turn capability. d) Two steered wheels with 360° turn capability [19].

The steered wheel control, is more precise than the differential steering and has smoother turning and can be used in all applications, even the towing. It can have one or two steer-drive wheels. See Figures 2.8 (b) and 2.8 (c). The Steer-Drive Three Wheel configuration in Figure 2.8 (b), works similar to a tricycle and it is most used on forklift AGVs [18]. In the other hand, the Two Steer-Drive Wheel configuration presents tight turning capability providing movement in most directions [18]. Finally, the quad wheel configuration permits the AGV to have the capability to move in any direction, providing the highest degree of maneuverability and the center of the robot always stays in the center of the guide path. This provides advantages to environments where space constraints may be a problem [18]. The scheme of the configuration can be seen in Figure 2.8 (d).

Path Decision

AGVs must be able to make decisions during their route. There are different methods to aid the AGVs to make the decisions. If the AGV system is non-wire, the path select mode is used. In the other hand, if its a wire guidance system, the frequency select method is used.

- Path select

  In this method, the AGV changes its path by selecting one of the pre-programmed paths. This selection is simpler, the robot selects a path based on one of the programmed paths [20] and follows the path based on the map stored in the AGV memory.

- Frequency select

  Frequency select works by analyzing the frequencies emitted from the floor. As an AGV approaches an intersection, it detects the different frequencies on the floor and decides the best path by comparing to the data stored in its memory [20]. The difference in frequencies are required only at intersections.

2.2.1 System Monitoring

Monitoring the AGV system becomes important, as the complexity of system increases, to prevent slowdown or even a breakdown in the system. The sooner the identification of the problem, the better. There are three approaches [20]:

- Locator Panel;

  This approach is a simple monitoring system where the panel shows if a vehicle is in the area obstructing the guide path, however, it does not specify which vehicle and its status [20]. A light near the area where the vehicle is can be lighted up to display that an AGV is there. Also, timers can be used to monitor if there is a problem with an AGV between tasks.

- CRT Color Graphics Display;

  This method is a usually a real time monitoring type [20]. Therefore, it can detect problems, identify the AGVs, show the location of the events (failures, vehicle moving, blocked by traffic, loaded or empty, vehicle destination and battery status) in a graphic display.

- Central Logging and Report.

  This approach is similar to the previous one, however a list of every vehicle and its conditions is presented in a table, such as, location, destination, condition, mode of control, load status and alarm condition [20]. This method is preferred when it is necessary to make log reports over time about the system performance, thus, making it possible to print and analyze the vehicle historical data and enhance the system efficiency.

2.2.2 Vehicle Dispatch

Four methods can be used for vehicle dispatch, a brief description of them are presented below:

- Onboard Dispatch Selector;

  This method consists of a control panel equipped in each AGV which permits an operator to dispatch the vehicle by selecting a specific task. This is the most common system management and the most flexible type [20].

- Off-board Call System;

  This method varies in complexity according to the need. Ranges from simple types, which involve only a push button at a call location, to stop the passing guided vehicle. Additionally, call terminals can be implemented, which not only can call a specific vehicle, but can also remotely dispatch that vehicle to other destinations without the operator interface [20].

- Remote Dispatch Terminal;

  This approach is used when the complexity of the problem does not justify the use of a totally computer-controlled implementation, however it is not simple enough for simple solution. A remote dispatch is used in systems to have a centralized dispatching control of the AGVs. Therefore, the operator must know the vehicles position and condition to dispatch efficiently the right tasks to the right vehicles. This remote terminal is usually a CRT graphic display or locator panel. This method performs better when a high degree of selective movement with automatic load transfer is required [20].

- Central Computer Control.

  This approach is the highest level of control possible. The vehicles answer to the central computer's commands for its tasks and positions [20]. This is an automated version of the previous mentioned Remote Dispatch Terminal. According to Savant Automation [20], most integrated material handling systems incorporate central computer control for AGVs system management. If the volume is high and the system has automatic load transfer, this management is very efficient. This efficiency happens because, when a load is ready to be picked up, the central computer selects the closest unoccupied vehicle to pick up and send to its destination [20].

There are cases where is advantageous to mix the methods presented above together in one solution. This depends on the type of control offered by particular vendors, but in many cases they can be operated in several different modes at the same time [20]. It is good to use as a backup system, e.g., if the main one fails for some reason, you can use a backup system to continue the management until the primal system is restored. It is important to note that not all central computer systems permit this type of approach [20].

2.3 Path Planning and Obstacle Avoidance

The main idea is that the AGV will leave its ordinary task when a gas hazard is detected and seeks for its origin. For this to be possible, the robot will need a path planning method to reach the leakage position. In this way, according to A. Gasparetto [21], path planning algorithms can be defined as algorithms that generate a geometric path, from an initial to a final point, passing through pre-defined via-points, either in the joint space or in the operating space of the robot. These algorithms can be divided in three types [21]:

- Road map techniques;
- Cell Decomposition methods;
- Artificial Potential methods.

2.3.1 Roadmap techniques

The roadmap techniques are based upon the reduction of the N-dimensional configuration space to a set of one-dimensional paths to search, possibly on a graph [21]. In a subtle way, it maps the free space into one dimensional curves. The path planning consists into connecting the initial point to the final point along those curves. As Figure 2.9 shows, the roadmap, called visibility graph (a graph where all nodes are the obstacles vertices), can be associated with a graph.

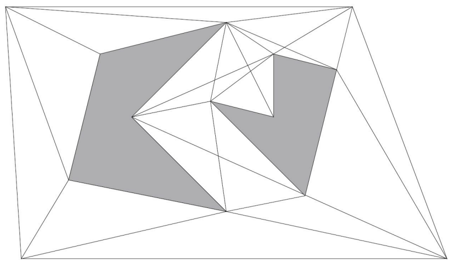


Figure 2.9: Visibility Graph [21].

In Figure 2.9, the grey areas are the obstacles and, to find the optimal path between the initial and final point, the graph can be searched for the best combination, which is generally the shortest path.

There are other roadmap algorithms based on Voronoi Diagrams, which are made by the paths being generated with maximum distance between of all obstacles. Therefore, Figure 2.10 presents this behaviour.

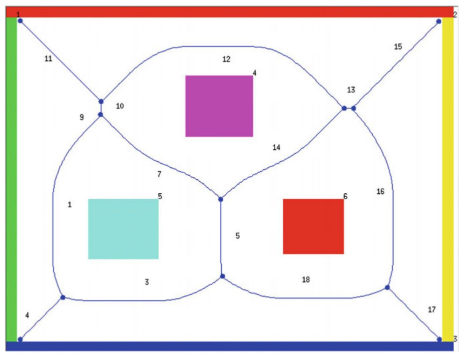


Figure 2.10: Voronoi Diagram [21].

2.3.2 Cell Decomposition Methods

This type of method consists of dividing the free space of the robot into several regions, which are called cells. Then, the path between adjacent regions is easy to generate. To obtain the path between the initial point and the goal, it is necessary to obtain the adjacency paths between the regions. The path planning problem is, again, turned into a graph searching problem, and can therefore be solved using graph-searching techniques. [21]. Figure 2.11, demonstrates the exact cell decomposition method step by step.

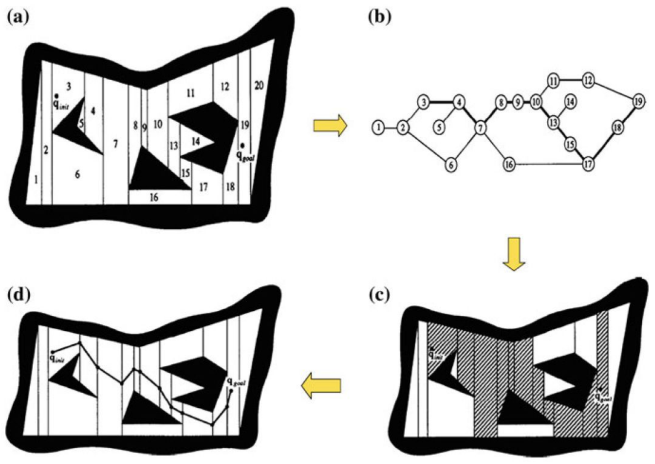


Figure 2.11: Exact cell decomposition: a) subdivision of space into numbered polygons, b) connectivity graph, c) regions to be crossed, d) path [21].

2.3.3 Artificial Potential Methods

The techniques based on these method consist in making the free space be subjected to a potential field generated by the goal point. Thus, the robot is affected from the same field. In this sense, the free space represents the attractive potential field and obstacles generate a repulsive potential field. The sum of these two contribution is the total potential, which can be seen as an artificial force applied to the robot, aimed at approaching the goal and avoiding the obstacles [21]. Figure 2.12 demonstrates this behaviour.

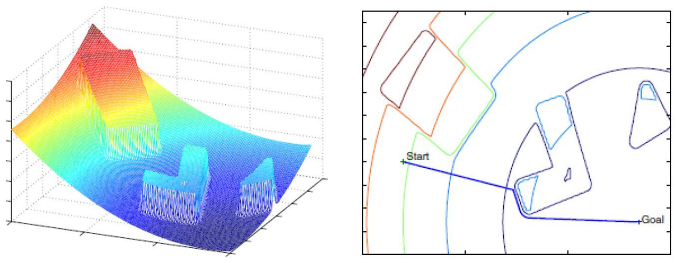


Figure 2.12: Potential field algorithm representation [3].

Path planning was developed by several researches along the years. Some of them are described here.

The classic Travel Salesman Problem (TSP) is always used to demonstrate the logistics of the best possible route for a seller to visit all n cities, without returning to those where he has passed and returning to the first city when the last city is visited [22]. Another classical problem of path planning is demonstrated by J. T. Schwartz and M. Sharir [23], which deals with the navigation of a three-dimensional body among known obstacles, called Piano Mover's Problem. In the latter case, the movements of the robot are defined as off-line planning, this is because the path planning is completed before the final execution of all movements. The problem of the Piano Mover's still inspired other works, such as that of the Sofa Mover's and the Generalized Mover's Problem [24].

The region filling method is demonstrated in [25], where the process of determining collision free paths is done through flat surface fills, with the implementation of computer graphics. In [26] a solution is identified for the kinematics of the redundant robots, that is, in robots that have high Degrees of Freedom (DOF), the complexity of the path planning also increases. Another approach to the high DOF is demonstrated in [27], i.e, it is possible to search the trajectories with the connection of local minimums of the potential function defined in the configuration space of the robot. Shortly thereafter, E. Ralli and G. Hirzinger [28] optimized this same algorithm through calculations of solutions with lower estimated execution time. With the implementation of a vision-based algorithm, M. Ollis and A. Stentz [29] shows how it is possible to guide a harvester by tracking the line between cut and uncut crops. From a stored museum map, the mobile robot called RHINO served as a tour guide being fully autonomous. The algorithm needed to be able to locate itself by comparing the readings of the map stored with the sensing applied in the robot's structure [30]. A small vehicle designed to carry two people is presented in [31], developing a solution with a pre-guidance to the problem of non-holonomic restriction, that is, robot without the ability to slide sideways. With the developments of the Bézier curves, the robots can realize curves in an autonomous way [32]–[34] .

With the obstacle detection method from the Vector Field Histogram (VFH), it is possible to identify unknown obstacles and at the same time direct the mobile robot towards the target [35]. The algorithm had some improvements with the works [36], [37], in which the trajectory is generated more uniformly and with greater reliability in Vector

Field Histogram Plus (VFH+), and in Vector Field Histogram Star (VFH*), the direction of the robot is verified guiding it around an obstacle.

Regarding probabilistic methods, the concept of reducing the complexity of the free space configuration is demonstrated by [38]. Still, because of the need to reconstruct the whole graph this method can not be applied in dynamic environments. This problem is addressed by some alternatives of this method, such as: Medial axis Probabilistic Roadmap (PRM) [39], Visibility based PRM [40], Lazy PRM [41] and sampling based roadmap of trees [42]. In dynamic environments, real-time path planning is presented in [43]. In this approach, the constraint methods along with the procedure of avoiding the local minimums are used as a strategy to ignore the obstacles.

In the simulation of this work, it was considered two of the most commonly applied algorithms for path planning. The first algorithm is A* [44], which uses the heuristic strategy to perform searches in graphs from the evaluation of the classification of nodes. Then, W. Y. Loong, L. Z. Long, and L. C. Hun [45] uses the algorithm to demonstrate in a Graphical User Interface (GUI) the functionality of a mobile robot moving from a starting point to a ending point. In [46], the navigation on a mobile robot in grid format maps shows the differences regarding the computational costs for several variations of A* algorithm. Also in the question of changing the base code of A*, P. Sudhakara and V. Ganapathy [47] adds the parameter of number of turns in the algorithm that the robot makes during the trajectory. Moreover, A* with Robot Operating System (ROS) is applied in [48] to create a global map in low cost robot movements.

Another common algorithm is the Rapidly Random Tree Star (RRT*), which calculates the paths by sampling free space with structures, that resembles a tree. This structure consists of several branches and nodes, generated by sampling randomly the free space [49]. In [50], the approach of producing smooth and viable paths through closed-loop results in easier paths and always connected to any pairs of states. In [51], it is identified that the RRT* planning process takes into account the kinematic constraints of the robot and the position of the obstacle to be diverted. Performance comparison in 3D spaces generated by Point of Clouds is made in [52], which demonstrates differences between computational time and costs of PRM and RRT* algorithms through simulations. A variant of Rapidly Random Tree (RRT), RRT-Rectangular, [53] determines that the algorithm has good performance in 2D path planning.

2.4 Mobile Robot Localization

Mobile robot uses similar techniques previously mentioned in the navigation section. Some of them are described in this section.

2.4.1 Laser Beacons

These methods work exactly the same way described in the AGV navigation through laser guidance (Figure 2.4) where the mobile robot have a rotating laser that tries to detect the reflectors position. To ensure the effectiveness of the method, it is necessary that the laser can detect and identify at least three reflectors without any obstacles between them [3]. Then, the robot position is computed by analyzing all the distances obtained from each reflector and the intersection made by the reflected laser beams.

2.4.2 Perfect Match

This approach is used in robot soccer competitions and industrial applications [3]. The method consists of several consecutive observations performed with robot's sensors such as, a camera. The data is optimized by numerical methods [54] and the result in a minimization of the errors. To improve the accuracy of localization, Kalman Filter is used combine the data from all the sensors [3].

2.4.3 Ultra-wideband (UWB) technology

This technology consists by one movable anchor and several fixed anchors in the environment that are integrated circuits which receives and re-transmits eletromagnetic waves with ultra-wideband frequency. The fixed anchors represent the inertial base with two or three Degrees of Freedom (DOF). The movable anchor keeps transmiting the radio waves and computes the distance to all the fixed anchors using Time of Flight (ToF). Then, with these data a trilateration algorithm estimates the robot's position. This techonology was already validated in [8].

2.5 Gas Search

This section will present the relevant information regarding gas leakage detection systems. For further discussions on detection methods, the reader is referred to [55]–[57].

2.5.1 Stationary Leakage Detection

In dangerous areas, stationary gas sensors are installed to detect gas leakages. Additionally, IR-Optical transducers are remotely operated and use passive or active IR spectroscopic scheme to gas detection. Depending on the optical configuration (short or long path, one or two path, diffuse background reflection or retroreflector) they can detect a plume locally or remotely in a range from 30 to 1000 m [58]. For instance, in pipeline leakage the temperature field disturbance in the leak proximity can be monitored. Naturally, the temperature disturbance occur by a pressured gas escaping and consequently, chilling out the air. In addition, detection of pressure changes in pipes is a possibility. Furthermore, acoustic transducers can be used to measure the noise that the fluid flow generates when escaping from a pipeline. For this purpose a sensor has to be placed, e.g, each 15 km [58].

2.5.2 Mobile Leakage Detection

Trained personnel with portable *in-situ* detectors or even trained dogs can be used to detect leaks by searching for gas leakages odors and noises. These detectors can be gas concentration sensors or even passive IR-thermography. The latter, is used to detect the temperature field disturbance on the leak site. Also, portable active IR-optical methods based on Tunable Diode Laser Absorption Spectroscopy (TDLAS) are remotely applicable detection methods for leaks in natural gas pipelines [58]. For airborne leak detection from helicopters or aircrafts, LIDAR systems with high-power lasers are applied [58].

2.5.3 Mobile Leakage Detection with Robots

Although remote sensing is advantageous because it permits rapid scans, is safer and easier to conduct, it is very expensive as see in [59]. Thus, the majority of mobile robots use *in-situ* sensors, e.g., [60]–[68]. While the direction of the wind should be considered in odor localization algorithms, e.g. [60], [61], this work will not consider the influence of the wind because the developed prototype is for indoor use. Generally, single-chemical detection is addressed, since the problem of measuring multiple-chemicals is not so common [59]. There are some co-operative multi-robot projects [69], even though the majority of the approaches are based on single robots concepts. There are several researches with odor source localization after a gas concentration threshold was passed, for instance: *E. coli* Algorithm, Zigzag/Dung Beetle Method, Plume-centered Upwind Search, Silkworm Moth, among others. As can be noted, several of them are inspired and tries to mimic the behaviour of many biological organisms. For an overview of several odor source localization algorithms, techniques, concepts such as chemotaxis and anemotaxis, the reader is referred to [70].

Studies and development in mobile robotics have steadily increased over the years. Of all the possible tasks a mobile robot can do, the ability to smell is a great way to detect gas leakage, that is, robots with a "sense of smell" [71]. Detecting gas leakage can also be performed by more than one robot, as L. Marques, A. Martins, and A. T. de Almeida [72] demonstrates by applying the Kalman filter it is possible to command five robots to find a certain concentration of gas.

The gas detection response is directly influenced by the discrimination of the gases to be detected, then M. Trincavelli [73] demonstrates an approach that inserts an array of sensors into the mobile robots. In this reasoning, each sensor is responsible for identifying a specific type of gas, speeding up the identification process. Another approach of multiple sensors to discriminate gases can be seen in [74], that develop a system with sensors that together distinguish different concentrations of propane, acetone, and ethanol. In order to detect large scale gas, P. Zhu, S. Ferrari, J. Morelli, R. Linares, and B. Doerr [75] applies the decentralized Gas Distribution Map (GDM) method. Generating a Hilbert map through probabilistic representations, addresses the task of finding gas concentration in the multiple classes. Therefore, it points to an alternative way to GDM to map an environment with measurements learned from the place. As the mobile robot advances into the environment, it is possible to automatically perform unsupervised learning through the system developed by H. Fan, V. Hernandez Bennetts, E. Schaffernicht, and A. J. Lilienthal [76]. In an on-the-fly way, the system discriminates the gases present in the site, which can serve as an extra tool for rescue teams.

Chapter 3

Robot@Factory Lite

Robot@Factory Lite competition appears as an opportunity to develop a low cost AGV that will meet the competition's goal as well as to implement the approach presented in this work. The idea is to cover the AGV working in a factory environment part of the main objective of this work. The autonomous gas search will be covered using a mobile robot that is described in Chapter 4. This was the ultimate reason that in this proposal two robots were used. The team from Instituto Politécnico de Bragança (IPB), IPB@Factory, returned victorious. The certificate can be seen in appendix A and the trophy in Figure 3.1. In this regard, the Robot@Factory Lite competition will be described as well as the AGV implemented, its features, control, hardware, platform and software worked on. In addition, the algorithms will be addressed.

Figure 3.1: Robot@Factory Lite 1st place trophy at Festival Nacional de Robótica 2019.

3.1 The Competition

The Robot@Factory Lite (RaF) competition have the objective to challenge and encourage students and researchers to develop an AGV that must pick boxes from an incoming warehouse, which represents the materials in a warehouse environment, and according on the type of the box, deliver to its corresponding location.

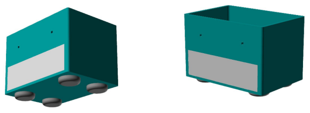


Figure 3.2: The Boxes [77].

As Figure 3.2 illustrates, the boxes have a rectangular grey area which is where a metal part will be put so the AGV, with an electromagnet, can grip. The dimensions can be seen in the rules competition [77] and it is not in this work's scope. Inside the box, between the little dots according to Figure 3.2, Radio Frequency Identification (RFID) tags are put for the AGVs distinguish the type of boxes.

In this way, there are three types of materials:

- Raw;
- Semi-processed;
- Processed.

In this reasoning, if one box contains a raw material, it be must taken from the incoming warehouse, processed by two machines (A and B) and finally delivered to the outgoing warehouse. Figure 3.3 displays the incoming and outgoing warehouses, the start area and the machines, A and B.

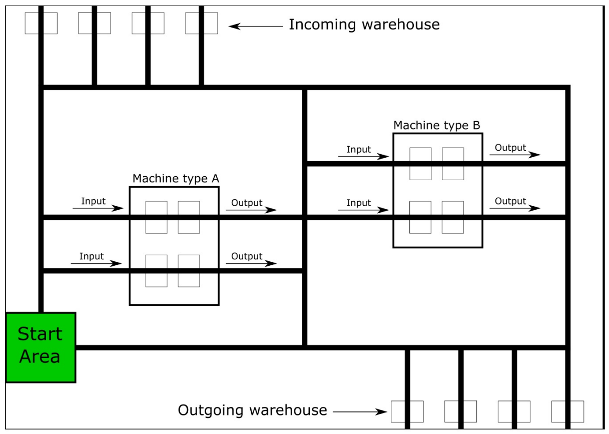


Figure 3.3: Robot@Factory Lite competition environment [77].

Note that, although the start area is on the southwest part of the area, it is possible to start in the northeast as the rules allow, and that the warehouse is symmetric both in X and Y axes (considering the origin at the middle of the area).

In the other hand, if a box contains a semi processed material, it must be processed just by machine B and then, delivered to the outgoing warehouse. By last, if the box contains a processed material, it just needs to be delivered to the outgoing warehouse. Therefore, if a raw material is processed by type A machine, it becomes a semi-processed material, and so on. The information on the type of the materials is summarized in Table 3.1.

Type of Box	**Destination**
Raw	Machine A
Semi-processed	Machine B
Processed	Outgoing warehouse

Table 3.1: Boxes and its destinations.

3.1.1 The rules

The competition is divided in two groups, one is for participants below 18 years old and the other, for above. It is organized in three rounds. The first round requires that the competitors just take the four boxes from the incoming warehouse and deliver to the outgoing warehouse without worrying about the type of the boxes, i.e, there is no need to check the RFID info as the boxes are all processed materials. Points are counted for each box correctly delivered within the outgoing warehouses delimitations. If there is a draw, the tie breaker is the time, i.e, the competitor that delivers the four boxes correctly in the shortest time win the first round. In the second round, some materials are going to be processed as in this case, semi-processed boxes are going to be present. In this regard, machine type B is going to be considered. Finally, in the third round, all three materials are considered.

The competitors AGVs are isolated 15 minutes before the start of the competition. It's possible to touch the robot only after the competition started in the respective competitors' turn. In each turn, the competitors have 10 minutes to test the robot before starting the round. In this interval, they can freely test the robot and the competition environment to make small adjustments. If the timer runs out or they decide to start their round, they can only start the robot at the start area, and after, can only touch the robot only if something goes wrong. Also, they can just re-attempt only if the time between tries is longer than one minute, limiting the attempts to less than 10, as the round time limit is 10 minutes [77].

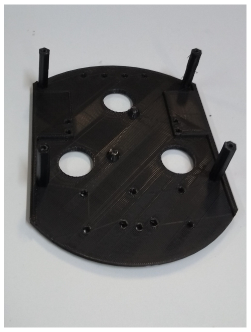
3.2 AGV's Hardware

As this robot is low cost, all the parts are simple and easy to understand. Table 3.2 summarizes its components.

Component	**Quantity**
DC motor	2
Wheels	2
Castor wheel	1
RFID reader	1
Line sensor	1
Electromagnet	1
Caster wheel	1
Battery support	2
Motors' driver	1
Arduino Uno	1
Arduino Uno shield	1
Lithium battery	2
Step down converter	1
Micro switch	1
Battery charger	1
MOSFET FDP3682	1

Table 3.2: Eletronic components and quantity.

The robot's base and the sensors supports were made in a 3D printer. Figure 3.4 displays the bottom and upper row of the robot's base, respectively.


(a) Bottom base of the robot.

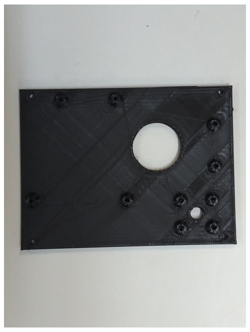

(b) Upper base of the robot.

Figure 3.4: Robot's upper and bottom bases.

The DC motors are without encoders because we did not have all the I/O ports necessary for this low cost project. Thus, the AGV control relied using timers as we did not have the capability to use odometry to aid the navigation. Figure 3.5 display one DC motor coupled with a wheel.

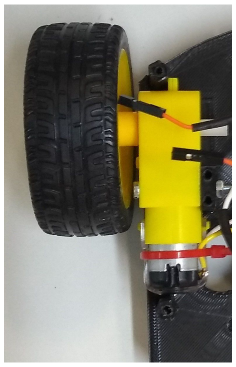
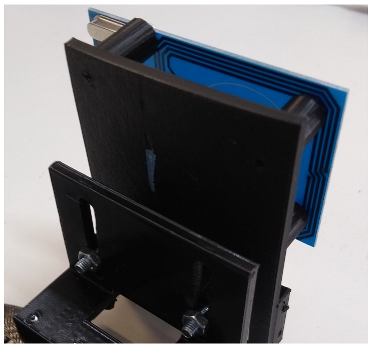
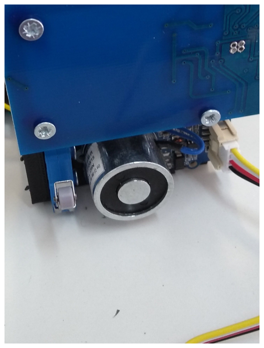


Figure 3.5: Left DC motor coupled with wheel.

In addition, the RFID reader used can be seen in Figure 3.6a coupled with its printed support.


(a) RFID reader (blue PCB) coupled with its support.


(b) Electromagnet and switch.

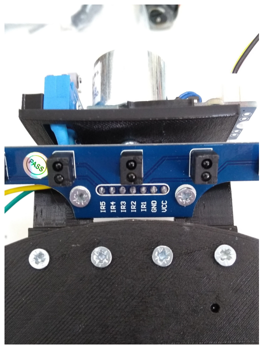

(c) Line sensor.

Figure 3.6: Robot's front components.

In front of the AGV, on the sensor supports, the line sensor, RFID reader, electromagnet and the micro switch were installed as can be seen in Figures 3.6b and 3.6c.

The robot was powered with two lithium 3.7 V batteries in series. As the output was 7.4 V and Arduino Uno can only endure 5 V in its input, the stepdown module was used to reduce the voltage to 5 V. Figure 3.7 shows the module.

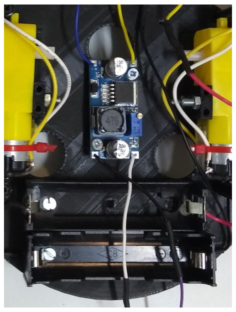


Figure 3.7: Stepdown module.

To control the AGV, an Arduino Uno was used. To facilitate the connections, an expansion shield was used. Figure 3.8 shows the Arduino Uno, its expansion shield, the dual motor driver module and the power switch.

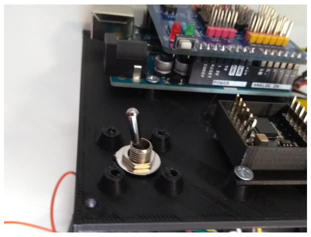


Figure 3.8: Arduino Uno, expansion shield, motor driver and the power switch.

An electric schematic was made to display all its connections. The connections can be seen in Figure 3.9.

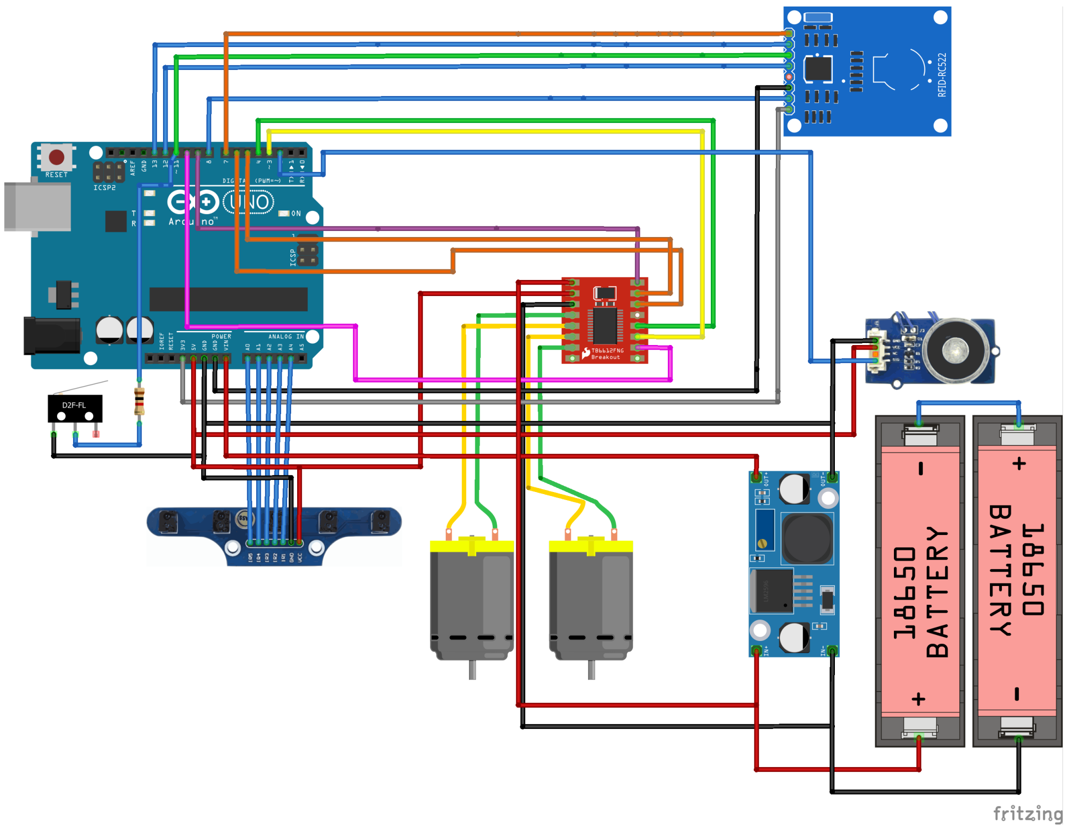


Figure 3.9: AGV's electric schematic. 1) Arduino Uno. 2) RFID reader. 3) Microswitch. 4) Infrared line sensor. 5) Motor driver. 6) DC motors. 7) DC-DC converter. 8) Lithium batteries. 9) Electromagnet.

Pin 11 connection in the microcontroller is shared between the micro switch and the MOSI pin from RFID reader, see Table 3.3. To ensure that both worked, a 1 kΩ resistor was put in series with the micro switch. In this way, when the AGV bumped into a box, the MOSI still worked as the resistor prevented pin 11 to be grounded by the switch. To aid the understanding, Table 3.3 presents the connections.

Microcontroller and its peripherals connections	
RFID	
SCK	13
MISO	12
MOSI	11
RST	8
SDA	7
IRQ	N.C
GND	GND
3.3 V	3.3 V
IR Sensor	
IR5	A0
IR4	A1
IR3	A2
IR2	A3
IR1	A4
GND	GND
VCC	5 V/VCC
MotorA - Left / MotorB - Right	
PWMA	9
AIN1	6
AIN2	5
PWMB	10
BIN1	4
BIN2	3
MOTORA	Left motor wires
MOTORB	Right motor wires
VCC	5 V/VCC
VM	<12 V / Batteries wires
GND	GND
Solenoid	
SIG	2
VCC	5 V/VCC
GND	GND
Microswitch	
Mid connector	11
One extremity connector	GND
Stepdown	
IN+	Battery+
IN-	Battery-
OUT+	5 V/VCC
OUT-	GND

Table 3.3: Microcontroller and its peripherals connections

Also, a reverse polarization protection was installed to prevent to damage the robot's circuitry. A MOSFET FDP3682 was used. Thus, the right schematic protection was used in Figure 3.10.

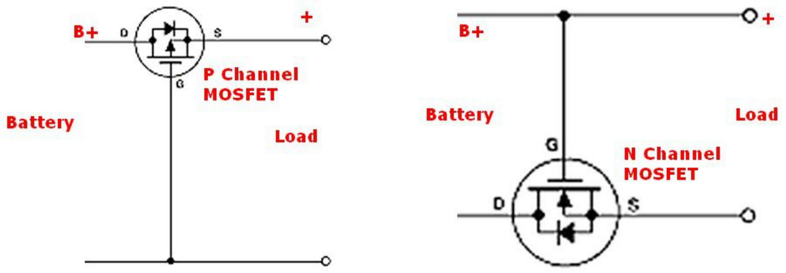


Figure 3.10: MOSFET reverse polarization protection diagram. B+ is the positive battery pole. Left schematic is for a P channel MOSFET. Right schematic is for a N channel MOSFET.

If a MOSFET P channel variant is used, left schematic can be used. The side view and the isometric view from the low cost AGV can be seen in Figures 3.11 and 3.12.

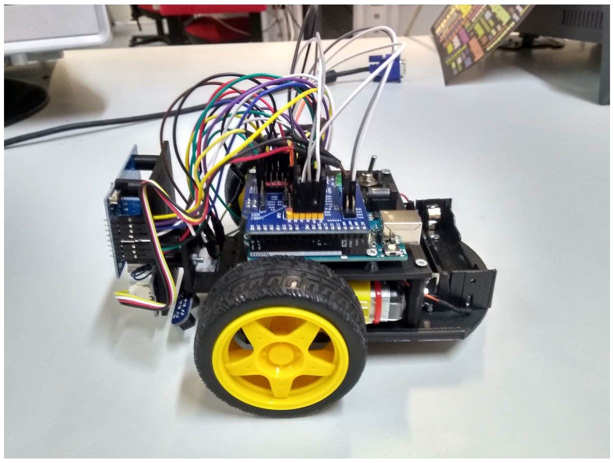


Figure 3.11: Side view of the low cost AGV for the competition.

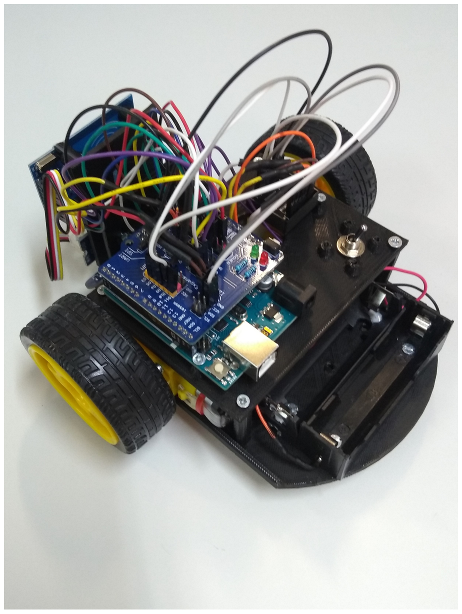


Figure 3.12: Isometric view of the low cost AGV for the competition.

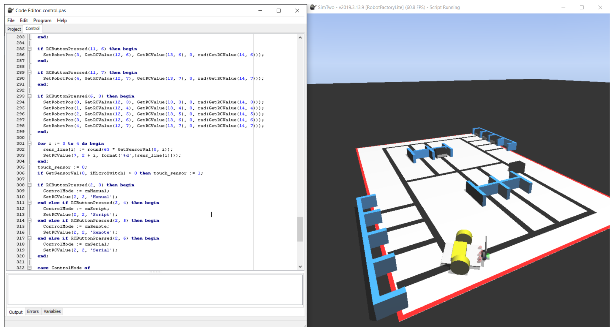
A manual was developed to describe the AGV's assembling and connection step by step. All the content can be see in Appendix D. The manual was developed mostly to help encourage other students, but anyone could use to participate in the competition.

3.3 Platform Simulation used and Software Solution

The competition provides a simulation environment in SimTwo simulator, with the characteristics close to the real scenario [77]. The simulation software presents a realistic 3D model of the robot and the environment with dynamic constraints similar found in the real scenario. In this way, the competitors can tests and validate their solutions to the challenge in a simulated environment before transitioning to the real scenario.

3.3.1 Platform

The competitors are free to develop the AGV which they find most suitable for the competition, as long as the robot's dimension obeys the competitions rules [77]. Not only the default competition robot's manual was been developed to help, see Appendix (D), but also the realistic model of the robot was made [78]. Also, they are free to edit the simulator scenario provided. The graphic part is made in XML and the control part in pascal language. The simulation environment can be seen in Figure 3.13.



```
  end;

  if RCButtonPressed(11, 6) then begin
    SetRobotPos(3, GetRCValue(12, 6), GetRCValue(13, 6), 0, rad(GetRCValue(14, 6)));
  end;

  if RCButtonPressed(11, 7) then begin
    SetRobotPos(4, GetRCValue(12, 7), GetRCValue(13, 7), 0, rad(GetRCValue(14, 7)));
  end;

  if RCButtonPressed(6, 3) then begin
    SetRobotPos(0, GetRCValue(12, 3), GetRCValue(13, 3), 0, rad(GetRCValue(14, 3)));
    SetRobotPos(1, GetRCValue(12, 4), GetRCValue(13, 4), 0, rad(GetRCValue(14, 4)));
    SetRobotPos(2, GetRCValue(12, 5), GetRCValue(13, 5), 0, rad(GetRCValue(14, 5)));
    SetRobotPos(3, GetRCValue(12, 6), GetRCValue(13, 6), 0, rad(GetRCValue(14, 6)));
    SetRobotPos(4, GetRCValue(12, 7), GetRCValue(13, 7), 0, rad(GetRCValue(14, 7)));
  end;

  for i := 0 to 4 do begin
    sens_line[i] := round(63 * GetSensorVal(0, i));
    SetRCValue(7, 2 + i, format('%d',[sens_line[i]]));
  end;
  touch_sensor := 0;
  if GetSensorVal(0, iMicroSwitch) > 0 then touch_sensor := 1;

  if RCButtonPressed(2, 3) then begin
    ControlMode := cmManual;
    SetRCValue(2, 2, 'Manual');
  end else if RCButtonPressed(2, 4) then begin
    ControlMode := cmScript;
    SetRCValue(2, 2, 'Script');
  end else if RCButtonPressed(2, 5) then begin
    ControlMode := cmRemote;
    SetRCValue(2, 2, 'Remote');
  end else if RCButtonPressed(2, 6) then begin
    ControlMode := cmSerial;
    SetRCValue(2, 2, 'Serial');
  end;

  case ControlMode of
```

Figure 3.13: Robot@Factory Lite simulation environment.

Even though the simulator is accurate, it does not take into account the microcontroller limitations such as the processing speed and memory. In this case, it is also provided a Hardware in the Loop (HiL) tool as well [78]. With this tool, the user can insert the robot's microcontroller in the simulation loop. In this regard, the simulator process the information coming from the Integrated Circuit (IC), which are the motors' speeds and, soon after, sends the robot's sensors (line sensors readings, RFID tags and microswitch detection) data back to the IC. After this, the IC processes the data and sends newer data to the simulator, and this is repeated again. For this reason, the competitors can implement their codes that will control the real robot and perform the tests in the simulator. The SimTwo HiL loops takes 40 ms to occur. The HiL system diagram can be seen in Figure 3.14.

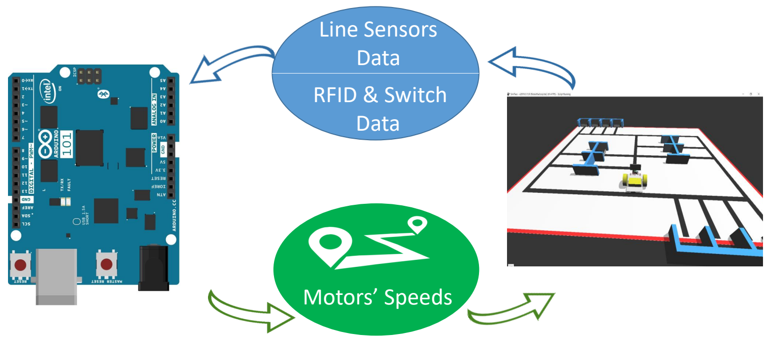


Figure 3.14: Hardware in the loop system diagram, adapted from [78].

Figure 3.15 displays the AGV model and its characteristics. Note that, it is faithful to the real AGV in relation to the dimensions.

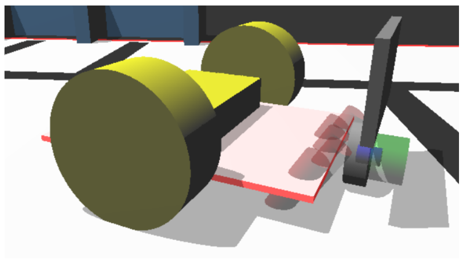


Figure 3.15: AGV model representation in the simulator.

Moreover, the AGV sensors are modeled as well. The five red cylinders in Figure 3.15 represents each infrared sensor in the real robot. The green cylinder represents the electromagnet and the grey rectangular prism represents the RFID reader. With all these tools, the transition between the virtual and real world is simplified, which permits more flexibility to the competitors.

3.3.2 Robot's control

To the AGV follow the competition lines, a simple P controller is used. A relative position to the center of the line is computed. Using the fixed width of the line used in the competition and the center as origin, the relative position is computed with the five line sensor readings. Note that all this control is encapsulated and provided for the competition users with high level routines. The routines are “followLineLeft”, “followLineRight”, “followLine”, “turnRight”, “turnLeft” and “moveRobot”. These routines just ask for a value for the speed of the robot and a proportional K value for the P control. With this, a linear velocity v is defined just with the nominal speed valued passed to the routine. An angular speed ω is defined by $-K \cdot IRPos$. “IRPos” being the general name for three different relative positions of the AGV to the center of the line. For the “followLineRight” routine, the “IRPos” is a variable that contains the data regarding the relative position of the robot right of the center of the line. On the other hand, “followLineLeft” is the relative position left of the center of the line and, finally, for “followLine” is the distance from the center of the line from both sides. The rest of the routines are just high level functions that ask the v and ω in the case of “moveRobot”, and just ω for the turn routines. For the robot to know when to turn, there is an encapsulated routine that counts the intersection crosses between the lines. It works by detecting the line sensor saturation. If the AGV is following a line left of the center of it, some of the outputs of the line sensor will be out of the line, thus, having higher infrared reflectance and analog value (the competition area is white, see Figure 3.3, except the lines). In this reasoning, if a intersection comes, all the outputs analog values will be lowered, and a cross is counted. With the robot’s origin and the crosses between the lines, the user and the robot can know where it is between tasks. With these tools, the software solution for the competition will be addressed in the next subsection.

3.3.3 Software solution

In this subsection, the solution implemented and validated in the three rounds of the FNR 2019 competition are going to be described. The FNR 2019 is the biggest scientific encounter with robotic competitions in Portugal promoted by University of Porto, Institute for Systems and Computer Engineering, Technology and Science (INESC TEC) and by the Sociedade Portuguesa de Robótica, for further information the reader is referred to [79] and [80].

First Round

As explained in Section 3.1.1, in the first round there was no need to identify the boxes as the objective was just to deliver all the four boxes to their respective outgoing warehouses. Thus, for the sake of memory occupation comparison, the first round was solved using a Finite State Machine (FSM) algorithm. This was done, because the first round is the simplest of all the three and it is possible to implement the solution through this method. Figure 3.16 displays the idea behind the algorithm.

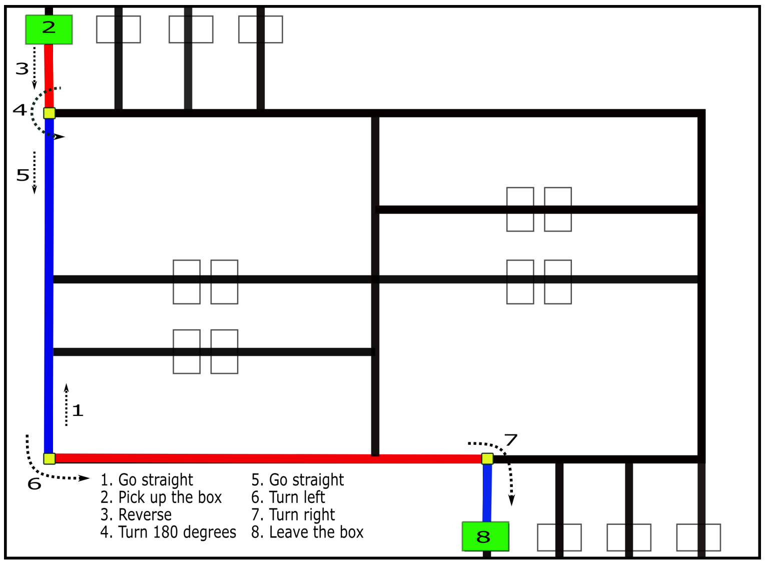


Figure 3.16: First box solution concept of the FSM.

Figure 3.16 displays, for the first box, eight states. Although the rest of the boxes used the same concept, more states were needed to solve the problem. For the second and third box, the robot needed to reverse from the position of the last box delivered and turn 90° to perform the mid route (shortest) to reach the next incoming warehouse box. After picking up the box, the robot would perform the same route through the mid lane to deliver it to the outgoing warehouse. However, for the last box, the route after picking it up, was the rightmost lane as this is the shortest path.

Second Round

The FSM method would not work for the second round forward. This happens because there are many possibilities and the microcontroller program memory would no suffice to cover all the cases. In other words, as there was the possibility of two types of box for each incoming garage, one type of box being a processed part and the other a semi-processed part, if the box was a semi-processed part the routes possibilities would increase largely. With this in mind, the similarities between the routes to deliver the two types of boxes needed to be observed. In this sense, a function was implemented that addresses these possibilities and similarities between the routes. To distinct the cases, the code is able to identify the type of box. In addition, the information needed were passed by parameters. This code were implemented in such a way, that it could resolve all the possibilities from the first and second round. Figures 3.17 and 3.18 display the similarities that needed to be observed and taken into consideration to make the implementation possible with the hardware that our team had for the competition.

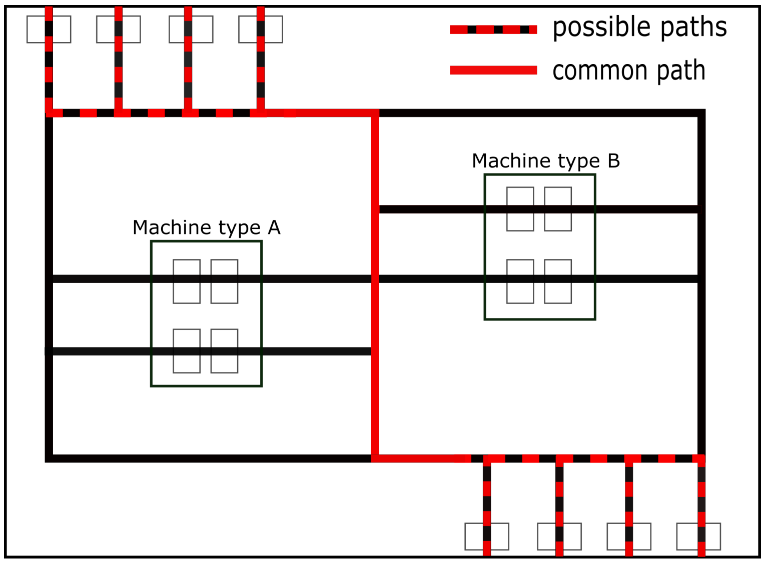


Figure 3.17: Common paths between the processed parts' routes.

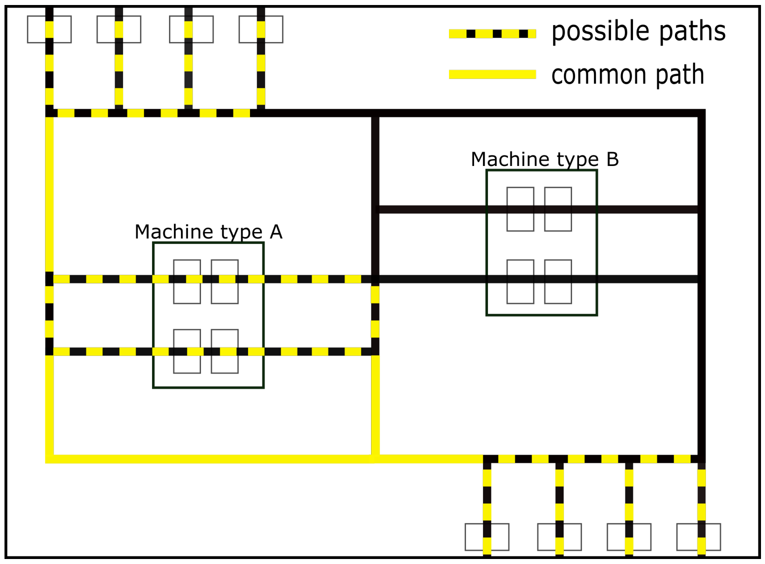


Figure 3.18: Common paths between the semi-processed parts' routes.

Note that in Figure 3.18, although the routes are for the semi-processed parts and they should go to machine B, in the competition, there was no specification in which machine the part should go in round 2. The only request was to put the box through one machine.

Third Round

For the third round, the algorithm needed to be updated. As mentioned before, in the third round all three types of boxes are introduced, and so the third is the raw part. This box needs to pass through the two machines before going to the outgoing warehouse. Therefore, the updated version followed the same concept as the second round idea. Figure 3.19 displays the commons paths between the possible routes.

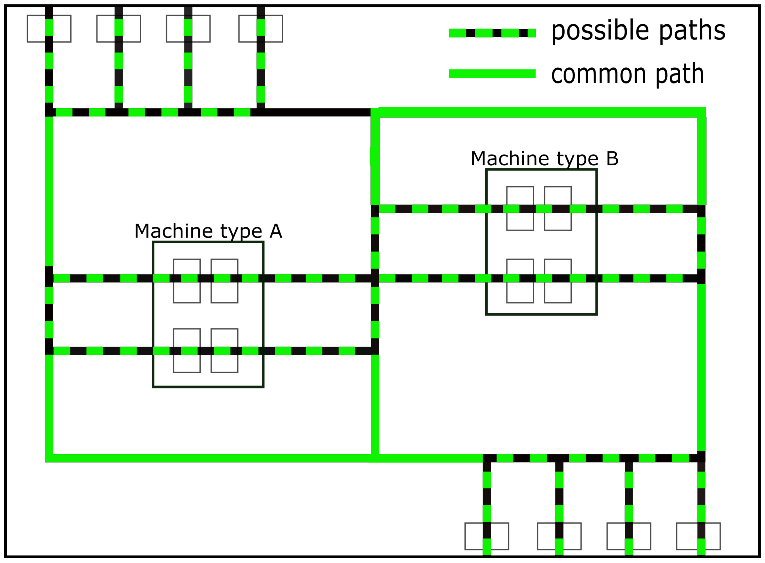


Figure 3.19: Common paths between the raw parts' routes.

Moreover, a pseudo-code was written to explain the idea of the solution. The general function example can be seen in Algorithm 1.

Algorithm 1 Main Function

```
1: function MAIN FUNCTION
2:     Go straight until touch the box
3:     $Box \leftarrow Part\ Type$
4:     if $Box == Processed\ Part$ then
5:         function ROUTE(1)
6:         end function
7:         function ROUTE(2)
8:         end function
9:     else if $Box == Semi - Processed\ Part$ then
10:        function ROUTE(3)
11:        end function
12:        function ROUTE(4)
13:        end function
14:        function ROUTE(5)
15:        end function
16:        function ROUTE(2)
17:        end function
18:    end if
19: end function
```

Note that, the main function identifies the boxes and which route needs to be performed. After this, it calls the route function that have all the possible paths. This function can be seen in Algorithm 2.

Algorithm 2 Function ROUTE

```
function ROUTE(int NumCase)
    switch Numcase do
        case 1
            Pick up the box and deliver to the outgoing warehouse
        case 2
            Go back to incoming warehouse
        case 3
            Pick up the box and put in the machine
        case 4
            Pick up the box in the machine
        case 5
            Drop the box in outgoing warehouse
end function
```

Following the examples above, if the box is a processed part, case one and two are run from the route function. In the same reasoning, if the box is a semi-processed part, the cases three, four, five and two are called.

Chapter 4

Hazardous Substances Detection

In the concept of this proposal, if there is a leak in the environment that the AGV is transporting hazardous materials, it will abandon its task and will search for the gas leak origin. Therefore, the AGV will behave as an AMR, i.e, will act as an Autonomous Mobile Robot and search autonomously for the gas source instead of just following simple programming instructions as an AGV. For this kind of behaviour, it is necessary sensors and processing equipment to perceive the environment and become autonomous as an AMR. In this concept, the WMR used in this work hardware will be described below, with the appropriate modifications.

4.1 Wheeled Mobile Robot's System Architecture

This WMR was used in a previous work [3], and it was modified to this work. The robot's system diagram can be seen below:

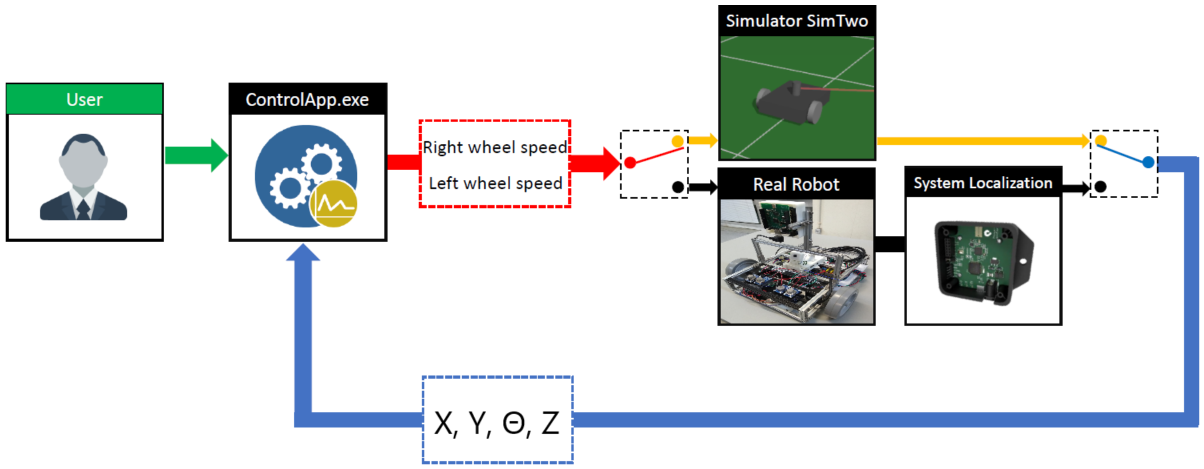


Figure 4.1: System architecture functionality in blocks diagram, adapted from [3].

It was developed a control application so that this robot on its own can search for gas leakages. Therefore, following the blocks diagram in Figure 4.1, a user can insert a waypoints so the control application running in a computer, will interpolate a route that will simulate the AGV working, all this concept will further explained in Chapter 5. The route insertion can be seen in Figure 4.2.

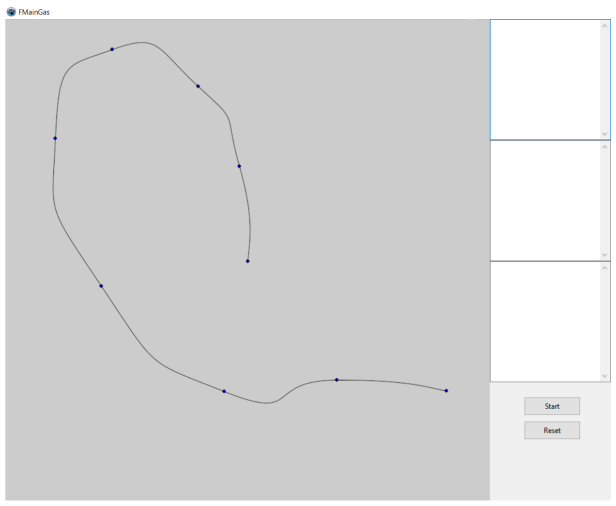
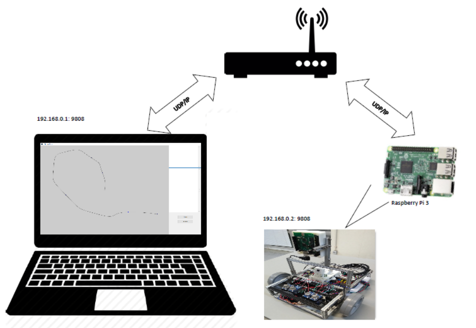


Figure 4.2: Control application.

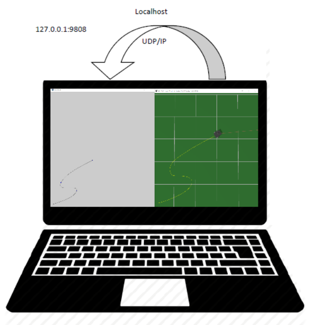
Soon after the user press start, the control app will sample the spline generated by distance and in this way, every 20 cm a new point is sampled and the left and right motor speeds are computed and sent to the simulator or the real robot by WiFi using

UDP communication protocol. The control app interaction is simple, the user can add waypoints using Left Mouse Button (LMB). For the first inserted way point, the app will interpolate a spline using the robot's position at the time of the insertion, which in Figure 4.2 is the center blue circle. The user can remove waypoints by using the Right Mouse Button (RMB). This can be done until all the waypoints are removed, or click on the reset button to clear all the points.

As the control app is running on a external computer, UDP communication protocol was chosen. This decision was made because TCP protocol is slow in comparison with UDP, the data is updated in real time as will be shown in Sections 4.1.2 and 4.2.1, and the control app itself checks the data sent, thus, there is no need to use TCP protocol to guarantee the data reception. The selection between the simulator and the real robot is done by adjusting the IP and port. The only requirement is that both the control app and the real robot or the simulator must be connected to the same network. The workflow of the communication can be seen in Figure 4.3.



(a) Real robot.



(b) Simulated robot.

Figure 4.3: Figure a) illustrates the communication workflow using the real robot and Figure b) displays the simulation communication workflow, adapted from [3].

Both the communications work similarly. The difference is, when the robot is being simulated, the simulator will communicate with the app, and if the real robot is being used, the app will communicate with the real robot instead as can be seen in Figure 4.3.

Additionally, the app is responsible for several developed algorithms in this work such as, all the obstacle avoidance algorithms, LIDAR and gas sensors data gathering and processing, trigonometry calculations and others. On the course of this thesis, all those information will be presented. There are some part of the work that was developed in MATLAB that communicates with the application, they will be described in Sections 4.2 and 6.3.

4.1.1 Robot's geometry and kinect model

The proposed WMR has a differential geometry, and, in this reasoning, has three degrees of freedom $[x, y, \theta]^T$, in the inertial basis seen in Figure 4.4. In this type of geometry, two wheels are independent and connected to DC motors. A third wheel, is used just for support and it is a swivel caster wheel. The reason why this type of geometry was used is because of its control and construction simplicity [3]. The kinect model will be similar as in [3]: first the pose will be described, then linear and angular velocities and finally the non-holonomic constraint model will be presented. Note that all the mathematical theory is going to be based on [2].

Robot's Pose

The robot has three degrees of freedom and is represented in a 2D space by a state vector in a global referential. Therefore, Figure 4.4 displays the robot's referential in an inertial basis (denoted by the (X_I, Y_I) axes) which is the global referential

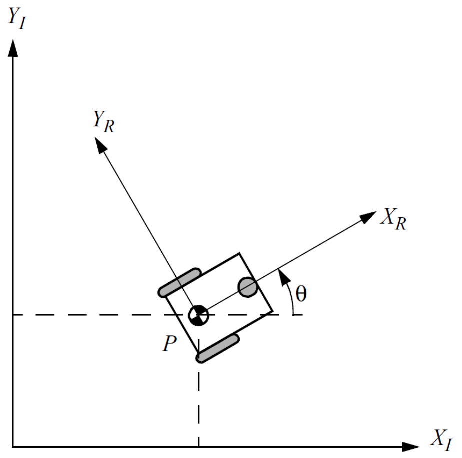


Figure 4.4: Robot's referential in a global reference frame, adapted from [2].

$$\xi_I = \begin{bmatrix} x(t) \\ y(t) \\ \theta(t) \end{bmatrix}, \tag{4.1}$$

where equation (4.1) represents the global space vector in the inertial basis.

The robot's referential (X_r, Y_r) seen in Figure 4.4 is related to the global referential in equation 4.1, by rotational matrix $R(\theta(t))$

$$R(\theta(t)) = \begin{bmatrix} cos(\theta(t)) & sen(\theta(t)) & 0 \\ -sen(\theta(t)) & cos(\theta(t)) & 0 \\ 0 & 0 & 1 \end{bmatrix}, \tag{4.2}$$

$$\xi_R = R(\theta(t)) \cdot \xi_I. \tag{4.3}$$

Linear and Angular Velocities

To obtain the robot's velocities, first the robot's state space reference frame must be derived. Thus, the speed model is displayed

$$\dot{\xi_R} = \begin{bmatrix} \dot{x(t)} & \dot{y(t)} & \dot{\theta(t)} \end{bmatrix} = \begin{bmatrix} cos(\theta(t)) & 0 \\ sen(\theta(t)) & 0 \\ 0 & 1 \end{bmatrix} \cdot \begin{bmatrix} V(t) \\ \omega(t) \end{bmatrix}. \tag{4.4}$$

As the differential robot geometry has two independent wheels, each with a diameter d, given a point P, the robot's overall speed is in function of

$$\dot{\xi_R} = \begin{bmatrix} \dot{x} \\ \dot{y} \\ \dot{\theta} \end{bmatrix} = f(l, d, \theta, \dot{\varphi_1}, \dot{\varphi_2}), \tag{4.5}$$

where l in Equation 4.5 is the euclidean distance between the robot's center of mass and P, θ is the robot's angular position and finally, $\dot{\varphi_1}$ and $\dot{\varphi_2}$ are the spinning speeds of each wheel.

The system inputs are the linear V and angular ω velocities of the robot, both variables can be obtained by the wheel speeds which are demonstrated by

$$V(t) = \frac{V_l(t) + V_r(t)}{2}, \tag{4.6}$$

$$\omega(t) = \frac{V_r(t) - V_l(t)}{b}, \tag{4.7}$$

where $V_l(t)$ is the speed of the left wheel, $V_r(t)$ the speed of the right wheel and b is the wheel radius.

Nonholonomic constraint

To understand better this concept, for example, there is no lateral movement to park a car, so several maneuvers are used to park. In this work scope, Figure 4.5 displays the idea.

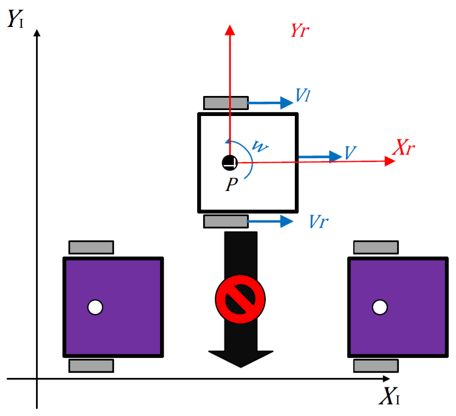


Figure 4.5: Nonholonomic constraint example in this work's scope, adapted from [3].

Therefore, wheels are typical sources of nonholonomic constraints. Nonholonomic systems are systems where it has at least one non-integrable (nonholonomic) constraint [81]. Considering our robot and its state space vector global frame in Equation 4.1. The robot's wheels can only move forward or backwards, i.e, there is no velocity perpendicular to the direction of the wheels. Assuming this, the kinematic constraint (a constraint is defined kinematic if only coordinates and velocities are considered), can be defined as:

$$\dot{x} \cdot sin(\theta) - \dot{y} \cdot cos(\theta) = 0. \tag{4.8}$$

Note that Equation 4.8 is non-integrable which makes our differential robot a non-holonomic system.

4.1.2 Robot's Electronic Hardware

The WMR was used in [3] and, modifications were implemented to fulfill this project. The hardware is installed in a non-conductive plastic base. The structure of the robot which was provided by my thesis mentors, is built using aluminum profiles with an X shaped cross section as can be seen in Figure 4.6. The profiles were bought from MakerBeam [82]. The idea behind this type of material is to provide robustness to the robot as it does not rust, it is light, rigid with high mechanical resistance and easy to prototipate.

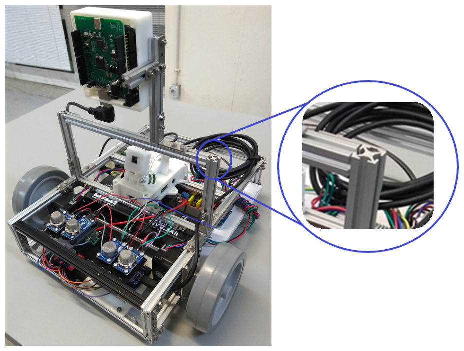


Figure 4.6: Cross section detail from the robot's profiles of the WMR.

The robot's dimensions can be seen in Table 4.1.

Robot Description	**Dimension**	**Unit**
Width	0.280	m
Length	0.350	m
Wheel diameter	0.100	m
Wheel thickness	0.025	m
Robot mass	3.710	Kg

Table 4.1: WMR dimensions, adapted from [3].

For the WMR to be autonomous and navigate in space, it is necessary sensors, localization systems, a central computer, microcontrollers, motors, batteries and integrated circuits. The electronic schematic of the robot can be seen in Figure 4.7.

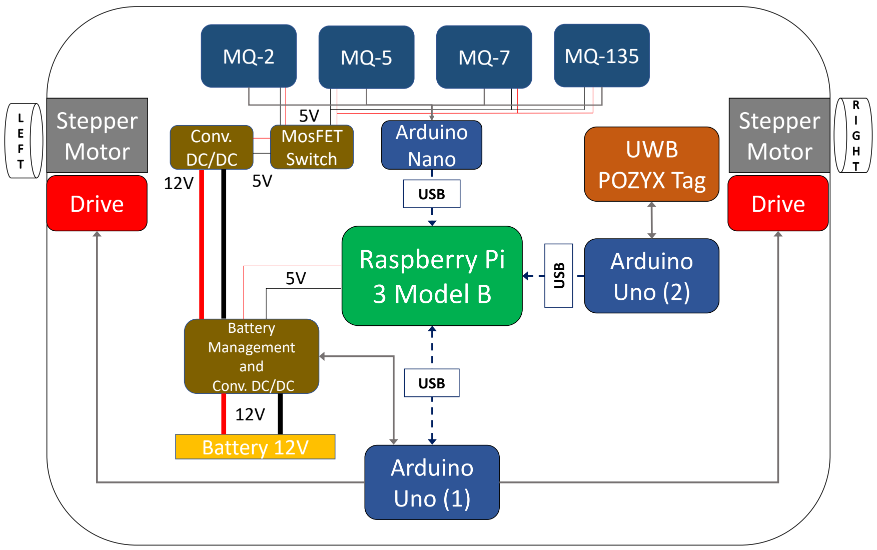


Figure 4.7: WMR robot's electronic schematic, adapted from [3].

Raspberry Pi

This computer is the mastermind behind all the other components of the robot. All the information is sent to it or from it. The Raspberry Pi 3 Model has a quad core 1.2 GHz 64 bit CPU, 1 GB RAM, wireless LAN and bluetooth connection, 4 USB ports, 40 general purpose I/O pins, HDMI, cameras, among others. Its main purposes are to run an app for performing Kalman Filter calculations, communicating with the control app in Figure 4.1 using Wi-Fi, and communicating with the two Arduino Uno (1), (2) and the Arduino Nano in Figure 4.7, processing their information. The Raspberry can be seen in Figure 4.8.

Figure 4.8: Raspberry Pi 3 Model B illustration [83].

Arduino Uno (1), (2) and Nano

Following Figure 4.7 concept, Arduino Uno (1) is connected to the CNC shield V3 and to the Allegro MicroSystems-a4988. The latter being the micro stepping driver to the motors with overcurrent protection. Arduino (1) is not only responsible for the motors control but also for odometry calculations [3]. The Uno specifications can be seen in [84].

Arduino (2), for instance, is connected to the Ultra Wide Band (UWB) Pozyx tag. Therefore, it receives the data from the Pozyx anchors which are going to be described further in this section and performs the trilateration calculations. As the official site declares [85], their product is a Real Time Locating System (RTLS) which has ultra-wideband, providing robustness against interference and has a precision of 10 cm. It can also penetrate thin walls. The tag can be seen in Figure 4.9.

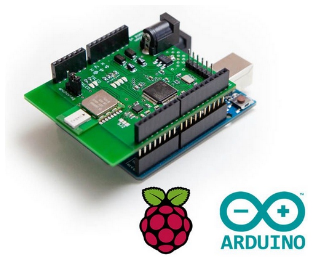


Figure 4.9: Pozyx illustration (upper PCB) [85].

Finally, the Arduino Nano, which was added in this work with the gas sensors, is to gather the gas data from all the four gas sensors simultaneously and to process it. To process the data, a low-pass filter was implemented to remove the sensors' noise and smooth out their response. The Nano IC is similar to the Uno technical specifications as both are based on the ATMega 328p chip, however the Nano is a more compact build and has more analog inputs and digital I/O pins. Since the power consumption of the module sensors is considerable and the autonomy of the autonomous robot is limited, a Mosfet switch (IRF520N) is used to control the power supply for the sensors. A Nano illustration can be seen in Figure 4.10.

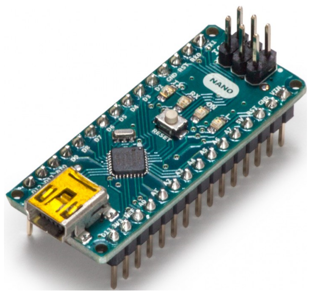


Figure 4.10: Arduino Nano illustration [86].

Gas Sensors

The objective of this work is to find a gas leakage and not the concentration values of the gases. Thus, low-cost sensors were bought for this purpose. It is important to declare that the sensors values shows just an approximate trend of the gas concentration in an error range. To obtain the real concentration values, it requires more precise and costly detection instruments and not just low cost gas sensors. Therefore, four gas sensors were embedded in the robot to obtain a variety of common toxic gases. The electronic schematic showed in Figure 4.7, demonstrates that all the sensors have simple circuit drives. In addition, they all have fast response and high sensitivity with a wide detecting scope. The embedded sensors can be seen in Figure 4.11.

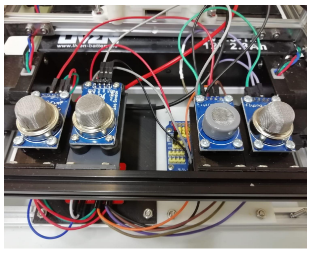
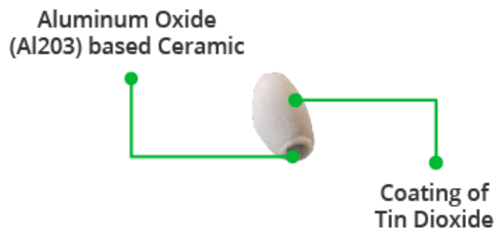
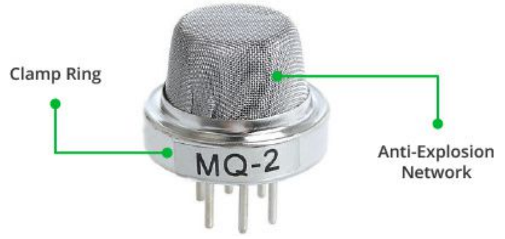


Figure 4.11: Gas sensors illustration.

The sensors works by heating a coating of tin dioxide, see Figure 4.12a, inside the anti explosion network displayed in Figure 4.12b. When the air is clean, donor electrons in the tin dioxide are attracted toward the oxygen particles and they are absorbed on the surface of the sensing material preventing electric current flow. If the sensor is in presence of toxic reducing gases, the surface density of oxygen particles absorbed is reduced. This allows electric current flow [87].



(a) Sensing element inside the anti-explosion chamber [87].



(b) Anti-explosion network [87].

Figure 4.12: Gas sensor structure.

First, the MQ-2 gas sensor is recommended for domestic and industry uses as it is suitable for detecting Liquefied Petroleum Gas (LPG), i-butane, propane, methane, alcohol, Hydrogen and smoke. Moreover, the MQ-5, has a high sensitivity to LPG, natural gas and town gas. In addition, it has low sensitivity to alcohol and smoke providing robustness against alcohol and smoke fumes. On the other hand the MQ-7 sensor has a high sensitivity for Carbon Monoxide (CO) gas. This gas is extremely toxic for any living being that uses hemoglobin as an oxygen carrier. Further on, is a tasteless, odorless and flammable gas. This gas common sources in today lives are from car exhaustion, stove and home wood fires. Finally, the MQ-135 module is for detecting Ammonia (NH3), Nitrogen Oxides (NOx), alcohol, benzene, smoke and Carbon Dioxide (CO2). It is important to note that these gas sensors detect a variety of gases and this is normal for this low cost gas sensors. They are not intended to indicate which gases are detected, as there are several reducing gases. Naturally, this must be used in an environment where such gases can be found.

As mentioned in Section 4.1.2, a second order low pass FIR filter was implemented. This was done to remove the sensors' noise, and the concept is just the take the low frequency variations of the data, i.e, the average values of each gas sensor output. For the filter project, the cutoff frequency was 342.96 Hz and the Nyquist frequency was 1000 Hz. The prescaler division factor set to the ATMega 328p Analog to Digital Converter

(ADC) was 128, providing 125 kHz of input clock frequency to the Nano ADC module. This was chosen to have a maximum resolution as the ATMega 328p datasheet states. In this regard, the impulse response of the filter and the frequency response can be seen in Figure 4.13.

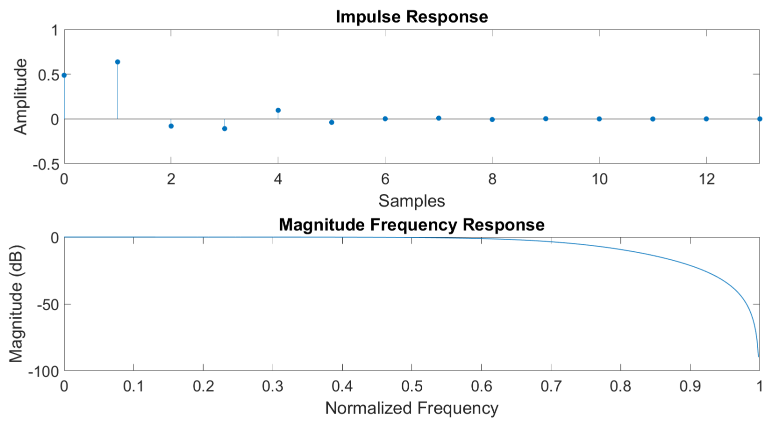


Figure 4.13: Impulse response and frequency response of second order low pass FIR filter.

The step motors and their drivers

The robot uses two NEMA 17HS16 step motors with 1.8° resolution. With the CNC Shield V3, the Arduino Uno (1) communicates with the two Allegro MicroSystems A4989 motor drivers to control the motors. Figure 4.14 displays the devices.

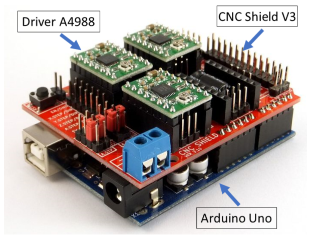


Figure 4.14: The motors drivers connected to the Arduino Uno (1) via CNC Shield V3 [3].

UWB Pozyx locating system

For Radio Frequency (RF) techniques, L. Piardi [3] states that there are two approaches well accepted by the academic community, the Received Signal Strength (RSS) and the

ToF. The first being computed by measuring the signal power at the receiver and the second is the principle of our localization system, which will be described in this Section. In addition, according to L. Piardi [3], there are other solutions using RF technology such as: Wi-Fi with an average error of 3-5 m, RFID with an average error of 2 m, Bluetooth with an average error of 1.5 m and UWB with an average error of 10-20 cm.

The UWB Pozyx system was chosen for the real robot for its precision and low cost. This device is a low cost RTLS that can report the position of a movable object within 10 cm precision and is suitable for challeging indoor environments [85]. The system consists of one movable tag, which is the transmitter, and fixed anchors, which are the receivers and have a limit of 1000 anchors per gateway. The more anchors, more the precision and slower the operating frequency. However, this is for business systems. For personal use, a max of eight anchors is recommended. In this work four anchors were used. The Pozyx system works by the movable tag (which is embedded in the robot by Arduino (2), see Figure 4.9), seen at the top of Figure 4.6, sending UWB radio signals to the fixed anchors. With the time the signal takes to reach the anchors, and the speed of the radio waves (which are very close to the speed of light in our atmosphere), the Pozyx system can estimate the distance between the movable tag and the anchors ($D = c * ToF$). After this, it uses a multilateration algorithm to position the movable tag in the environment. One of the four fixed anchors can be seen in Figure 4.15 powered by a 2000 mAh battery.

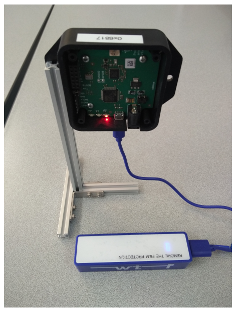


Figure 4.15: Pozyx tag being powered by a battery.

The principle is that a position of an object in n dimensions can be estimated by using $n+1$ Time Difference of Arrivals (TDOA). In our case, we have four fixed anchors. Thus, we can estimate a three DOF object. The multilateration is performed by computing all the TDOA and known locations of the fixed anchors. Thus, transmitter pose can be estimated. In other words, if the robot had one DOF and two fixed anchors, it would transmit UWB signals to both the anchors. With the TDOA and the anchors pose, it is trivial to estimate the robot's pose relatively to the anchors. The estimated position could be only in three regions displayed by Figure 4.16.

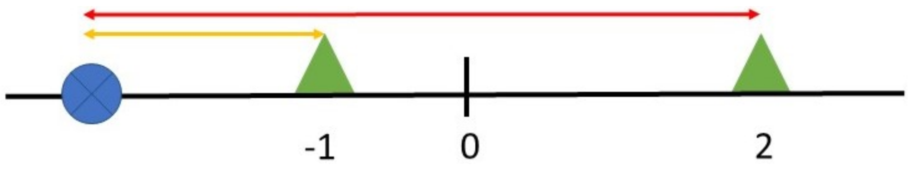


Figure 4.16: The unidimensional multilateration process illustration. Blue circle with cross represents the mobile robot pose and the green triangles the anchors with known pose.

The three possible regions in Figure 4.16 could be between the two anchors, to the right of the right anchor and to the left of the left anchor. Following the concept of TDOA, after each anchor retransmitting the signal transmitted by the movable tag, the distance would be computed for all fixed anchors. After that, the robot's pose would be estimated based on all these information (anchors pose and distance of the robot between each anchor). Thus, Figure 4.17 displays the concept that the robot's will use.

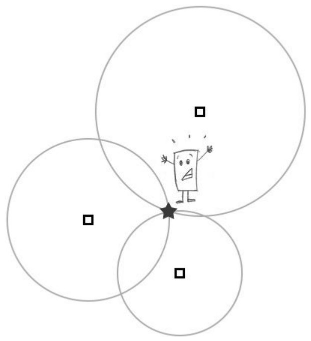


Figure 4.17: Pozyx trilateration illustration [85].

Note that in Figure 4.17, there are only three anchors and in this work it will be used four. As our robot will not change its elevation, the fourth anchor will be used just to increase the precision on the estimation. The creator Pozyx system (personal use), as Pozyx [85] states, provides 60 Hz of operating frequency divided by the number of fixed anchors.

Laser Ranger Finder

This type of device measures the distance of an object by emitting a laser beam. Although other techniques can be used for more precise results, normally ToF technique is used to measure the distance. In this work, a simulation model of the Hokuyo URG-04LX was used. This simulation model was done and validated through [88]. This model can be seen in Figure 4.18. In the previous work [3], this Laser Ranger Finder (LRF) was used just as a ground truth to validate the UWB localization system of the robot. However, in this work the LIDAR will be used to implement dynamic obstacles avoidance algorithms in an unknown environment, as the robot's localization system has already been validated.

Figure 4.18: Hokuyo URG-04LX [89].

The important technical specifications are presented in Table 4.2.

Characteristic	**Description**
Power source	5 VDC±5% (USB Bus power)
Measuring area	20 to 5600 mm(white paper with 70 mm×70 mm), 240°
Accuracy	0.06 to 1 m : ±30 mm, 1 to 4.095 m : ±3%
Angular resolution	Step angle : approx. 0.36° (360° 1,024 steps)
Scanning time	100 ms scan
Weight	Approx. 160 g

Table 4.2: Hokuyo model URG-04LX-UG01 technical specifications [89].

Regarding Table 4.2, this information was used to model the LIDAR in the SimTwo Simulator, using the accuracy, the measuring range and the angular resolution. Figure 4.19 displays the distance and angle measuring coverage. The LRF will be better explained in Section 4.2. Moreover, it can be noted that this device is suitable only for indoor use, as the ambient illuminance affects its functionality as can be seen in [89]. As briefly mentioned early in this topic, this type of LRF works by emitting a laser beam that is reflected by an object back to the sender. A photo diode receives this data and normally after a squaring circuit, the pulses are compared and the time difference between the two is used to compute the distance of the emitter to the obstacle.

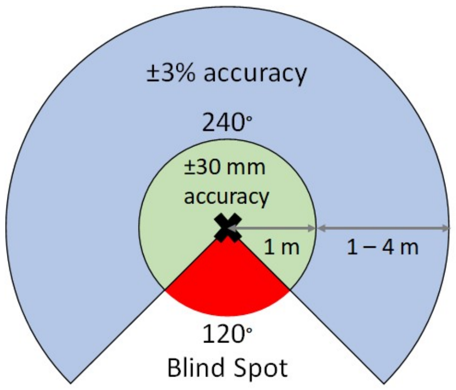


Figure 4.19: Hokuyo URG-04LX measuring area.

Robot Software

The robot's software was developed in pascal language in the Lazarus development environment. This software runs in the Raspberry Pi mentioned in Section 4.1.2 and is the main module of the robot. In addition, it manages the peripheral communications of the Arduino Uno (1), (2), Nano and the control application. Moreover, the software is responsible for the odometry and Kalman Filter calculations. As the Kalman Filter calculations is not on the scope of this work as this already came with the robot, its function will briefly described. Therefore, the only important information is that the Kalman Filter is responsible for the sensor fusion coming from the low cost localization system Pozyx, and the odometry, estimating the robot's pose in the environment [3]. This pose value is encoded and transmitted to the control application to be further processed.

4.1.3 Odometry

Also known as dead reckoning, odometry is a form of using data from motion sensors to estimate the robot's position at certain time. It uses a reference point and estimates the robot's position using previous information of the movement and orientation, i.e, the robot's pose $[x, y, \theta]$. It can be used for WMRs and for legged robots. For WMRs, measuring the rotation of the wheels using enconders in the motors, and knowing the circumference of the wheels, the direction of the movement, the speed and distance can be estimated. As the robot's locating system, Pozyx just informs the magnitude value of the robot's pose, odometry is essential to provide the robot's direction. Dead reckoning is widely used in robotics for its low cost and high sampling rate with circumstantial good accuracy. However, besides the advantages, as the mathematical model will demonstrate, it accumulates error through integration. In this sense, this method can not be used alone, because the values can deviate by several orders of magnitude depending of several circumstances. As L. Piardi [3] states, there are two types of errors for odometry, systematic errors and non systematic errors. Systematic errors can be:

- Wheels with different diameters;
- Wheels dimensions used in the calculations being different from the real ones;
- Wheels misaligned;
- Wheels deformation by defect or excess weight.

On the other hand, non-systematic errors are errors that can not be avoided, just mitigated and happens between the interaction of the robot and the floor:

- Irregular floor;
- Sliding and skidding of the robot;

- Motor slip due to acceleration, this happens if the inertial forces are greater than the rotational forces, generating error in the odometry.

In this concept, the mathematical model is trivial if the linear and angular velocities are constant at the sample time. The robot's control is simple and has a constant linear and angular speed per convenience, which will be explained in Section 5.3. Thus, the model can be solved by Euler method [90]. Integrating the kinematic model stated in Equation 4.1

$$\begin{aligned} x(t) &= \int_0^t v(t)cos(\theta(t))dt, \\ y(t) &= \int_0^t v(t)sin(\theta(t))dt, \\ \theta(t) &= \int_0^t \omega(t)dt. \end{aligned} \tag{4.9}$$

Solving Equation 4.9 assuming linear and angular velocities constant at the sample time is trivial by Euler method

$$\begin{aligned} x(k+1) &= x(k) + v(k)T_s cos(\theta(k)), \\ y(k+1) &= y(k) + v(k)T_s sin(\theta(k)), \\ \theta(k+1) &= \theta(k) + \omega(k)T_s, \end{aligned} \tag{4.10}$$

where T_s being defined as the sample period. Putting Equation 4.10 in a state space format

$$\begin{bmatrix} x(k+1) \\ y(k+1) \\ \theta(k+1) \end{bmatrix} = \begin{bmatrix} x(k) + v(k)T_s cos(\theta(k)) \\ y(k) + v(k)T_s sin(\theta(k)) \\ \theta(k) + \omega(k)T_s \end{bmatrix}. \tag{4.11}$$

4.2 Gas Search

In order to validate the approach and algorithms for the inspection and gas leakage detection, the real robot (already presented in this Chapter), already assembled, was taken as reference to develop the simulated robot in SimTwo [91]. Thus, with the environment created through the simulator, is possible to validate the approach for the detection of gas-emitting sources, and then, in a future stage of the work, to carry out the tests in real environment.

4.2.1 Simulated robot with LIDAR

The Hokuyo Laser Ranger Finder (LRF) was modeled in SimTwo and validated in [92], which presents noise similar to the real device, was used. In this way, it is possible to obtain results near to the real sensor. The real configuration setup was applied in simulation: number of laser beams, the laser position in the free space $[x, y, z]$, its angle in the free space, the angle coverage, beam length, sample period and the noise. To gather the data and use in the path planning, several trigonometrical computations were developed. First the projections of each beam angle were computed regarding the robot's referential as can be seen in Figure 4.20. After this, rotational and translation matrices were applied to the values to pass them from the robot's referential to the world referential, i.e, the simulation environment referential.

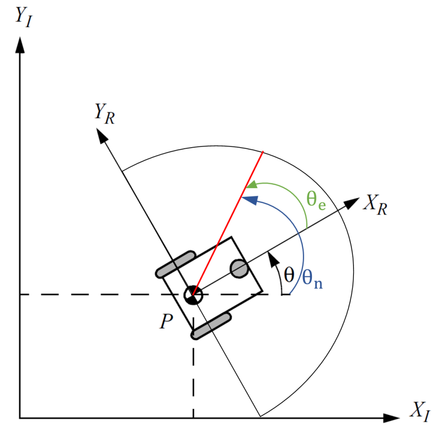


Figure 4.20: Robot's referential $[X_R, Y_R]$ placed in the world referential inertial basis $[X_I, Y_I]$. Red line is the laser from the LIDAR device, adapted from [2].

Using the same inertial basis stated in Section in 4.1.1, and with Equations 4.1, 4.2 is possible to obtain the obstacle position in relation to the world referential that is used in path planning

$$\xi_I = R(\theta(t)) \cdot \xi_R. \tag{4.12}$$

However, it is not sufficient to make the robot perform the routes without colliding with the obstacles, because the path planning does not take into account the robot's dimensions. Therefore, the Binary Heap A* is applied where 'ones' was used for obstacles and 'zeros' for free space. The idea is to enlarge the obstacles boundaries using binary dilation morphology [93]. Let A be a matrix which represents the path planning map and B an arbitrary square matrix with ones. The dilation can be denoted by

$$\begin{bmatrix} 1 & 0 & 0 & 0 \\ 0 & 0 & 1 & 0 \\ 0 & 0 & 0 & 0 \\ 0 & 0 & 0 & 0 \end{bmatrix} \oplus \begin{bmatrix} 1 & 1 & 1 \\ 1 & 1 & 1 \\ 1 & 1 & 1 \end{bmatrix} = \begin{bmatrix} 1 & 1 & 1 & 1 \\ 1 & 1 & 1 & 1 \\ 0 & 1 & 1 & 1 \\ 0 & 0 & 0 & 0 \end{bmatrix}. \tag{4.13}$$

Thus, using the dilation is possible to enlarge the obstacles. Thus, this can be used to consider the robot's dimensions in the path planning.

In addition as the simulator main loop takes 40 ms to update the robot's pose and the LIDAR takes 100 ms to update the information, a synchronization problem was needed to be addressed. If the gas search algorithm sent points which required the controller to compute high angular speeds (abrupt turns), the robot performed high angular speed turns generating errors in the obstacle mapping. This happens because the robot's pose updated in a higher speed than the LIDAR data, and the referential transformation calculations were jeopardized. To adjust this problem the controller needed to be configured for lower angular speeds, a linear interpolation with the robot's angle and new interpretation for the obstacles were done. The linear interpolation was performed by storing the last robot's angle and using actual robot's angle every 100 ms. Thus, abrupt changes in the robot's were compensated. For the new obstacle interpretation, to the path planning identifying an obstacle, instead of placing an 'one' in the matrix, it is added. Thus, the abrupt turn errors have low values in the matrix cells and the actual obstacles have high values. Therefore, the algorithm was adapted to identify as obstacles cells with high values. With this method, the desynchronization problem was mitigated.

4.3 Path planning and Gas Search Algorithm

To search for the gas leakage avoiding obstacles, not only the gradient search algorithm is needed but also a path planning. The path planning is used to reach the points that the gas search computed in an optimal way, without colliding with the obstacles. Binary Heap A* is utilized for the path planning, based on [94]. First, the user elaborates a trajectory by selecting waypoints in the developed control application. After, the application will interpolate a spline function with these points, as it can be seen in the next section. The SimTwo simulator sends the robot pose and the gas concentration, $[x, y, \theta, Z]_{Actual}$, to the application. The variables of communication between processes are presented in Figure 4.21.

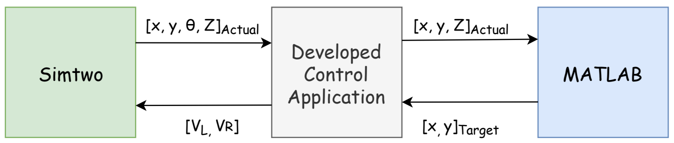


Figure 4.21: System Communication Structure.

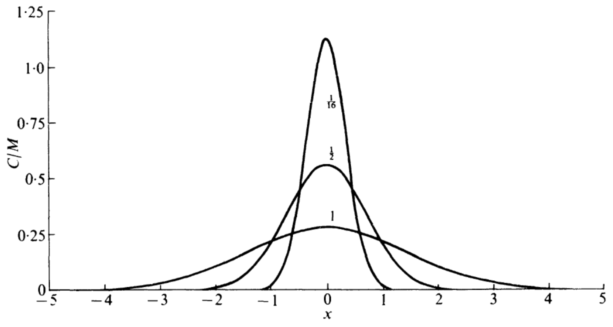
With this information, and sampling the spline function, the robot's control which will be described in Section 5.3, makes the robot to follow the trajectory computed by the interpolation. After the sampling and receiving the robot information, the euclidean distance between the robot actual position and the next sampled point is calculated. If the distance between these two points is above a threshold, the sampled point is kept to further calculations. If not, the next sampled point is tested. Thereon, the motors speeds (V_L, V_R) are calculated and sent to the simulator, in such a way that the robot will move to the target point. As the robot is moving, it keeps sending its pose data, and before the robot reaches the point, the application sends another sampled point and this cycle keeps happening until there's only one point left. In the last one, there's another distance threshold. If this criteria is met, the robot velocity will change from a constant speed to a decreasing linear speed that is weighted by the robot distance from the last point.

During all the process, the robot keeps measuring gas substances. In simulation, the system simulates the leak through a gas modeling. The gas concentration (Z) model for a constant diffusion can be seen in [95]

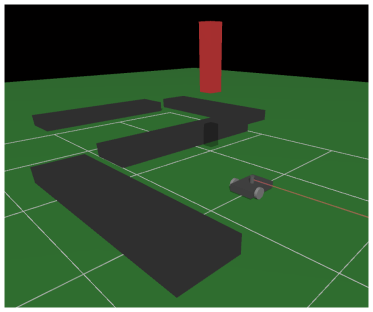
$$Z(x, y) = \frac{M}{2 \cdot \sqrt{(\pi \cdot D \cdot t)}} \cdot e^{-\frac{(h)^2}{4 \cdot D \cdot t}} \tag{4.14}$$

given by Equation (4.14) for the position of the robot (x, y). Where M represents the amount of substance deposited at time $t = 0$ at $[x0, y0]$; D is the diffusion constant; t the time and h is $\sqrt{(x - x0)^2 + (y - y0)^2}$, i.e, the euclidean distance between the measure position $[x, y]$, and $[x0, y0]$ the gas peak position.

As one can see, this model is time dependent, at each time the model represents another concentration value, see Appendix C for the time influence in the distribution. However, the robot converges rapidly to the peaks value. On the other hand, the diffusion is slow configured by parameter D. Thus the time influence was discarded by choosing a constant value for t. Figure 4.22a denotes the concentration values in one dimension representation, for several values of $D \cdot t$ with different x values.



(a)



(b) .

Figure 4.22: Figure a) represents the gas model in one dimension for several $D \times t$ and x values. Figure b) displays the simulation environment.

It is assumed an indoor environment so the influence of the wind is discarded. In future work, it will be addressed the wind disturbance as well as noise in gas acquisition. The simulated environment can be seen in Figure 4.22b where the red cylinder is used to represent the gas leak. Hereupon, the SimTwo will simulate the gas sensors measurements by calculating the gas concentration at the actual robot position. Note that, the gas leak source is simulated assuming a depressurized leakage, in this way, nothing will influence the shape of the distribution. If the concentration pass a certain threshold (defined by the user with range from 0 to 100), the system will discard the user defined trajectory, and will start to seek for the gas source.

The proposed algorithm for gas search is running on MATLAB and the communication variables can be seen in Figure 4.21 that uses UDP Ethernet datagrams. The application will send the robot pose and the concentration $[x, y, \theta, Z]_{Actual}$ to MATLAB that process the search algorithm for the leak localization. By this way, MATLAB sends the next position $[x, y]_{Target}$ with high probability of a higher concentration to probe. MATLAB was chosen to run the search tool, as is easier to implement more complex algorithms.

The MATLAB implemented search algorithm is described as follows. Let $[x, y]_{Actual}$

be the robot actual position and $[x, y]_{Previous}$ the robot previous position. Consider d the previous direction took by the robot, defined by $d = [x, y]_{Actual} - [x, y]_{Previous}$ and consider $\bar{d}$ the direction between the robot actual position and the position where was identified the higher gas concentration ($[x, y]_{Higher}$) and can be defined as $\bar{d} = [x, y]_{Higher} - [x, y]_{Actual}$. Define $Z_{Previous}$, Z_{Actual} and Z_{Higher} the gas concentration at $[x, y]_{Previous}$, $[x, y]_{Actual}$ and $[x, y]_{Higher}$, respectively.

If $Z_{Previous} \leq Z_{Actual}$, the robot target position will be defined as

$$[x, y]_{Target} = \begin{cases} [x, y]_{Actual} + d, & \text{if } Z_{Higher} \leq Z_{Actual}; \\ [x, y]_{Actual} + d + \bar{d}, & \text{otherwise.} \end{cases} \tag{4.15}$$

If $Z_{Previous} > Z_{Actual}$, the robot target position will be in the opposite direction, that means, if d is a horizontal movement then the new direction will be the opposite diagonal direction. In the same way, if d is a diagonal direction then the new direction will be the opposite horizontal direction. When the procedure take an opposite direction the choice to take right/left depends of the condition of Z_{Higher} is greater (or less) than the Z_{Actual}. In this way, soon after the robot quits the working trajectory, the robot will search for a direction with a greater gas concentration and will keep moving in the same direction until the gas concentration decreases. When this happens, it will search for the next direction where the gas concentration is higher, and so on.

The described algorithm takes the robot to a near location of the source. This algorithm makes no assumption of the exact gas model, denoted by 4.14, and will generate a set of target points and gas concentration measures with increasing concentration values.

Chapter 5

Navigation

In this chapter, the AGV's working route generation by the user will be explained. In addition, two path planning algorithms and one variant will be demonstrated. Finally, the robot's control will be stated.

5.1 Spline

As briefly mentioned in Chapter 4, the control application interpolates a opened spline where the user places the control points and continues doing so until the last point is placed. Splines can be classified in two types, the interpolation splines and the approximation splines. The first are the splines that pass through all the control points. On the other hand, approximation splines do not need to pass through all the control points, just the first and the last one, which are called nodes. As the robot that is going to be used, displayed in Figure 4.6, has a differential geometry and have non-holonomic constraints, the spline was chosen to make smooth curves.

After the user clicks the start button in the application (Figure 4.2), the spline generated is then sampled every 20 cm, see Figure 5.1.

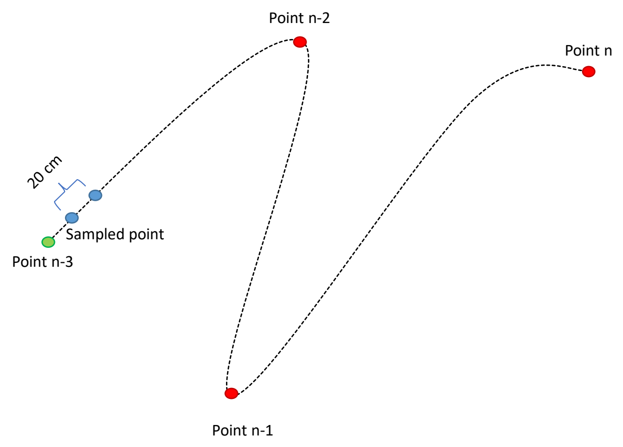


Figure 5.1: Sampled spline illustration.

Note that, points from $n-3$ to n are the control points to generate the spline. Point $n-3$ could be the robot starting point if the robot is stopped. The blue circles are the sampled points which are sent to the controller as the robot gets near the last point sent. This will be better explained further in Section 5.3.

5.2 Path Planning Algorithms

Three path planning algorithms were developed in this work, they are going to be described below.

5.2.1 A*

The A Star (A*) algorithm is one the most popular technique used in path finding graph traversals. It takes best of well known Dijkstra's and Best-First-Search algorithms into account. For Dijkstra, it takes the part that it favors the vertices closest to the starting point. On the other hand, for Best-First-Search, it takes the heuristics (favoring the cells that are closest to the goal). The algorithm works by choosing a node according to an f value, which is a cost value. This value is a sum of two other values, g and h, given by

$$f = g + h. \tag{5.1}$$

In this manner, at each step, it picks the node with the lowest f, i.e, lowest cost. The g value is the exact cost to move from the starting point to another cell on the grid, according to the path computed to reach there. In the other hand, the h value is an heuristic value, i.e, an estimated cost to move from the node that you want to reach and the goal destination node. This is called an "educated guess". There are several ways to compute the h and they are classified in exact heuristics and approximated heuristics. The first, which is time consuming, can be computed by calculating the distance between each pair of nodes before running the algorithm or the euclidean distance if there are no obstacles. On the other hand, the approximated heuristics, normally can be computed in three ways:

- Manhattan Distance;

  This heuristic is obtained by computing the sum of the absolute value of the coordinates differences, i.e., $h = abs(current\_node.x - target.x) + abs(current\_node.y - target.y)$. This heuristic is advised to use when the object is allowed to move only in four directions (north, south, west and east). This distance can be seen in Figure 5.2

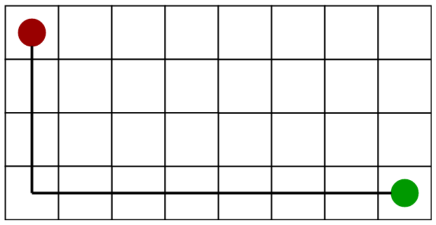


Figure 5.2: Manhattan distance, red circle is the starting point and the green the end point [96].

- Diagonal Distance;

  This heuristic is the maximum result of the differences of the nodes coordinates, i.e, $h = \max\{abs(current\_node.x - target.x), abs(current\_node.y - target.y)\}$. It is recommended to use when the object can move in eight directions (north, northwest, west, southwest, south, southeast, east, northeast). Figure 5.3 shows this concept.

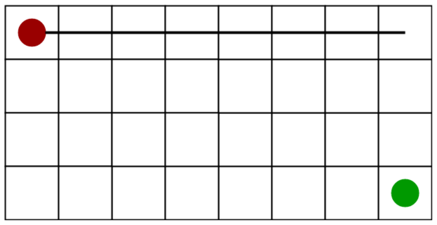


Figure 5.3: Diagonal distance, red circle is the starting point and the green the end point [96].

- Euclidean Distance.

  Finally, the euclidean distance is the straight-line distance between the two nodes. This heuristic is recommended when the object can move in all directions. Note that, the euclidean distance becomes an approximated heuristics when there are obstacles in the environment. In this sense, this heuristic was chosen for this work. Figure 5.4 shows the exact distance as there are no obstacles.

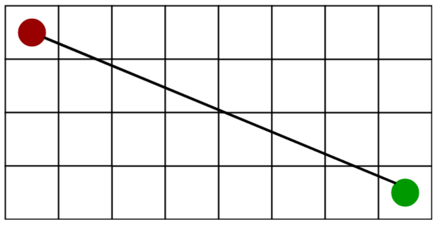


Figure 5.4: Euclidean distance, red circle is the starting point and the green the end point [96].

The A* uses open and closed lists just like Dijkstra's algorithm. The open set is a list of nodes that are to be explored and the closed set is a list of nodes already explored. In this way, as briefly mentioned in this Section, at each step the node with the lowest f is removed from the open set and the f and g values of its neighbors are updated accordingly, and these neighbors are added to the open set. The node removed from the open set, is added to the closed set. The stop criteria to the algorithm is when the open set is empty or the goal node has a f value lower than any node in the open set. Thus, to get the sequence of movements from the starting point to the goal is simple as A* algorithm keeps track of the predecessors nodes called parent nodes. In other words, each node has its parent node. Therefore, the sequence is just listing every parent node from the goal node until the start node. Figure 5.5 illustrates this algorithm.

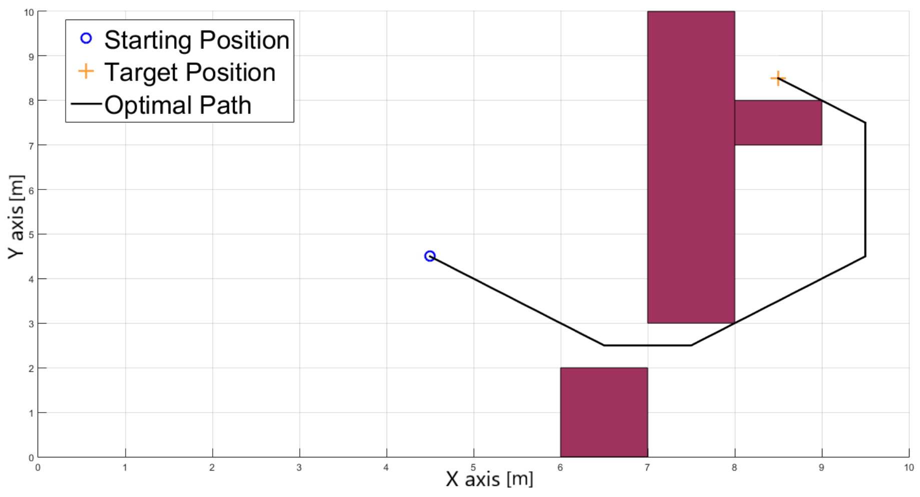


Figure 5.5: A* illustration

5.2.2 RRT*

To explain how RRT* works, it is necessary to explain how its predecessor RRT works. Rapidly Random Tree generates a tree by generating random nodes in the free space. It starts from the start node n_{start} and expands until it reaches the target node n_{target}. At each iteration the tree expands by generating a new random node n_{rand}. If this node is not inside a region that is considered an obstacle, then the nearest node $n_{nearest}$ from the tree is searched. If n_{rand} is reached without crossing any obstacles from the nearest node, taking into consideration the maximum step size, then the node is added to the tree and becomes the new vertex of this tree branch. Otherwise, if n_{rand} is reached without crossing any obstacles but with a distance greater than the step size, it returns a new node n_{new} by using a steering function, thus expanding the tree by connecting n_{new} with $n_{nearest}$ [97]. This n_{new} is generated at a Δq distance from $n_{nearest}$ in the same direction as n_{rand}, i.e, the new node is generated at an incremental distance between the nearest tree node and the random node that can't be reached. As described, collision checks are performed every iteration to ensure free connection between the new node and the nearest node, thus creating a tree in the free space. However, as the same way with other path planning algorithms, when implementing the algorithm with a mobile robot using dynamic constraints, the obstacles must be enlarged to consider the size of the robot as these algorithms does not take into consideration the size of the object.

RRT* for instance, works the same way as RRT. However, it introduced two promising features called near neighbor search and rewiring tree operations [97]. The first feature finds the best parent node for the new node considering a circle of radius r. It checks inside this circle the parent node with the lowest cost before inserting the three to become the new vertex of this branch. In other words, it connects the lowest cost neighbor to the new node. The latter feature, however, rewires the tree inside of the same circle to maintain the tree with minimal cost paths, i.e, optimizes the tree.

In this work a maximum step size of one meter per convenience is used. This was used as our cells is 1×1 meter. If the step size is higher or lower, the time for convergence to the target position was heavily increased or the goal was not found, taking into consideration a number limit of nodes. Two illustrations are presented below just to clarify the step size influence on the tree.

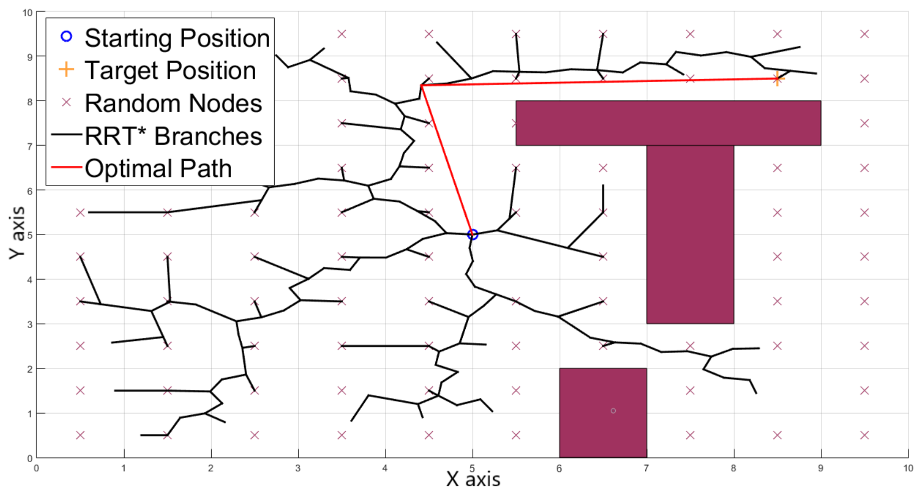


Figure 5.6: Tree expansion with a lower step size.

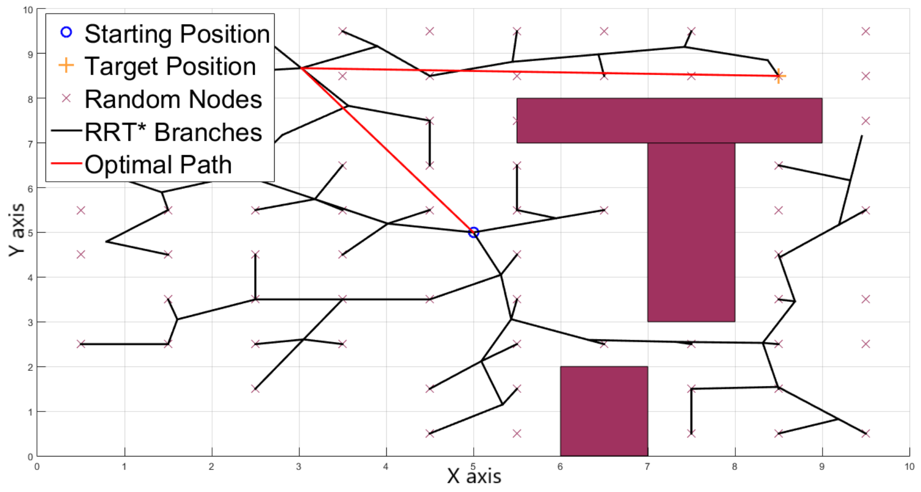


Figure 5.7: Tree expansion with a higher step size.

5.3 Robot's Control

The robot's control will not only act in the working route, but also during the autonomous search. In this reasoning, combined with the gas search algorithm described in Section 4.2, the obstacle avoidance algorithm will find a trajectory to reach the points sent by the gas search algorithm. Thus, the control developed in this work is a simple feedback control based on the points provided by the algorithms. They can be from the sampled spline for the working route, only the obstacle avoidance algorithm to reach somewhere or the gas search algorithm combined with the obstacle avoidance algorithm. The feedback control is chosen to mitigate the errors generated in the process, guaranteeing that the robot will follow the path. This was chosen instead of open loop controls, because they are not robust to model the errors [98]. In addition, as the robot is small (see Table 4.1), and it is just one part that moves, i.e, the robot can be considered as a moving square, the developed control can ignore the dynamics of the system. It works not only in simulation but also in a real scenario as this work will present in Chapter 6. This controller was based on [3]. The control is illustrated in a block diagram in Figure 5.8.

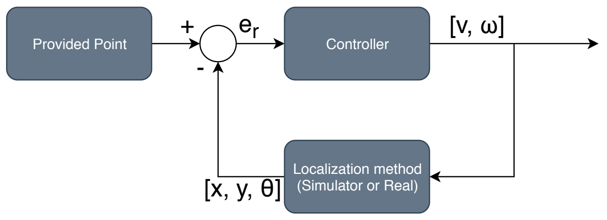


Figure 5.8: Feedback control block diagram.

As can be seen in Figure 5.8, the application sends a point (Provided Point) and this is summed with the actual robot's position in the global frame (Localization Method), this difference (e_r) in the pose is sent to the controller which calculates the robot's linear and angular speed to ajust its orientation to reach the point. Note that the actual robot's position in the real scenario is provided by the Kalman Filter Fusion, for more information see [3]. On the other hand, in the simulator there are functions which return the robot's actual position. In this concept, this cycle keeps going until the error e_r is below a pre-defined threshold. It is important to mention, as state before, as the robot has non-holonomic constraints, the robot can not have a linear speed in the Y_R direction (see Section 4.1.1). In this sense, the robot's linear speed is only to move front and back, i.e, in the robot's referential, only in X_R direction. Thus, to generate an angular speed, the combination of both the left and right motor speeds must be done to generate an angular speed (see Section 4.1.1).

This controller works well with the path plannings developed in this work as well as with the working route, because all of them prepares the path with an array of points in a way that the robot can follow the path smoothly.

As L. Piardi [3] states, the idea of a point following controller is to establish a line λ_n between the robot's pose $[x, y]$ and the next point of the trajectory $[x_n, y_n]$ where $n \in \mathbb{Z}_+$. After that, the controller calculates the difference of angles θ_e

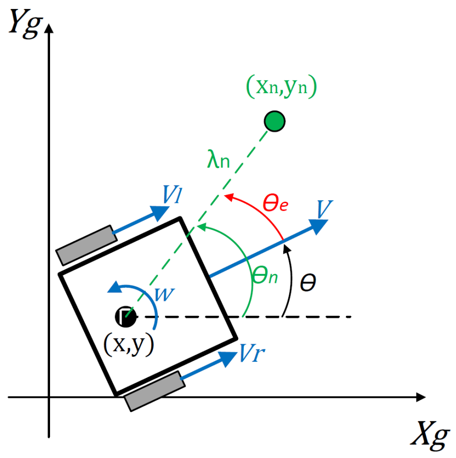


Figure 5.9: Mathematical illustration of the controller computations [3].

$$\theta_n = arctan(\frac{x_n - x}{y_n - y}), \tag{5.2}$$

$$\theta_e = \theta_n - \theta, \tag{5.3}$$

between the robot's orientation θ relative to the global frame and the λ_n orientation relative global frame θ_n. Figure 5.9 displays the idea. From Equations 4.6 and 4.7, V_r and V_l are isolated to obtain the values of the speed of the robot's wheels.

$$V_r = V - \frac{\omega \cdot b}{2}, \tag{5.4}$$

$$V_l = V + \frac{\omega \cdot b}{2}. \tag{5.5}$$

Note that in Equations 5.4 and 5.5, the left and right motors speeds are in function of the angular velocity ω and the robot's speed V. For V, it is assumed in the odometry calculations that the robot will be always at a constant speed during the sample time. Also, this was chosen because the robot searching for a gas leak in high speeds can disturb the gas plumes. Therefore, V becomes a constant value, V_0, that the robot must perform at all time. This can be seen if the arithmetic mean is computed in Equations 5.4 and

5.5, as the result is V. For ω, L. Piardi [3] used a proportionality constant K_{prop} times θ_e to provide its value

$$\omega_{ref} = K_{prop} \cdot \theta_e. \tag{5.6}$$

Thus,

$$V_r = V_0 - \frac{\omega_{ref} \cdot b}{2}, \tag{5.7}$$

$$V_l = V_0 + \frac{\omega_{ref} \cdot b}{2}. \tag{5.8}$$

To stop the robot, a deceleration function was made as can be seen in Figure 5.10, similar to [3].

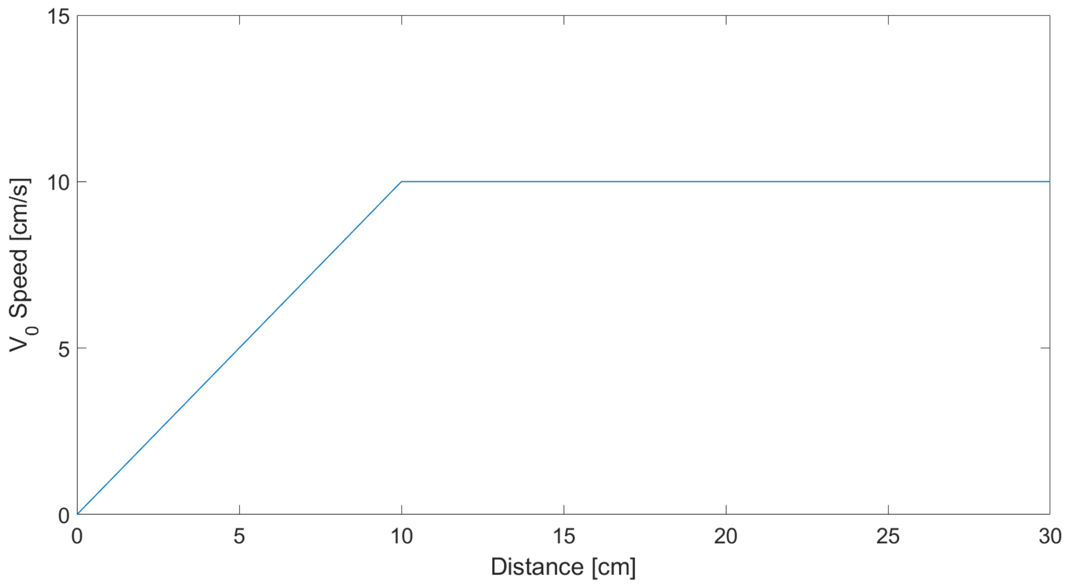


Figure 5.10: V_0 speed function.

In this way, when the robot approaches a certain threshold, which is 10 cm away of the last point, the controller starts to decelerate in a linear way until it stops. In other words, after it reaches the threshold, the speed is weighted by the distance of the last point.

5.3.1 Remote Control

The robot's control with buttons, in the robot's software, made by L. Piardi [3], was adapted to be remotely operated by the user using the keyboard with visual feedback. The camera can be seen in Figure 4.6 below the Pozyx tag.

Chapter 6

Results

In this chapter all the results obtained during this work will be presented. This chapter is organized as follows: First the competition results are described. Soon after, the working route performed by the robot in simulation is given. Next, the path planning algorithms performance comparison are displayed. Later, the obstacle avoidance results are presented. Thereupon, the gas search algorithm without obstacles results are shown. Then, the gas search algorithm with obstacles results are stressed. Also, the control's adaptation to remotely operate the robot is shown. Finally, a test with the real robot performing the working route gathering the gas data is displayed.

6.1 Robot@Factory Lite Results

As the AGV structure is simple and does not have an encoder, real tests in the competition proved to have a high dependency in non-systematic errors. Especially the irregular floor and sliding of the robot. This happened because without the odometry the code had one big flaw which was the time dependency. The actions such as: turn the robot 90°, 180° or even running the robot in reverse (soon after the robot took or dropped the box in a garage), were done with the help with the infrared sensor and the mean time that the robot took in several tests. However, with the non-systematic errors, the robot could turn in less or more time, resulting in errors in these actions, i.e, if the robot turned 90°

in less time, it would turn even more because of the time condition. Although, this flaw was known even before the assembly of the robot. It was decided to assemble the AGV without encoders. This happened because of little time to prepare for the competition and the desire to make a low budget AGV. Therefore, the competition results are displayed in Table 6.1.

Round	Best Time	Boxes
1	2:00’57	4 processed materials
2	3:48’80	2 processed and 2 semi-processed materials
3	2:58’00	1 raw material , 1 semi-processed material and 1 processed material

Table 6.1: Performed Times in the Competition.

In the second attempt of the first round the AGV managed to deliver all the boxes roughly in two minutes. However, this was in the second attempt. As we had ten minutes for the first round, three attempts were made, incrementing the speed at each attempt trying to reduce further the time. Other teams managed to deliver all the four boxes as well and the tie breaker was the time. Therefore, we lost the first round to another team in tie breaker. In the second round, our team was the only that could deliver all the four boxes. In the third round, during the third day, the teams faced intense competition as three teams could be champion. Our team had punctuation advantage, as we only needed to deliver three boxes to become champions. And so we did as our proposed algorithm proved to have superior performance to the other teams during the last round. However, the AGV could not deliver the last boxes because of the non-systematic errors. Figure 6.1 displays the competition area.

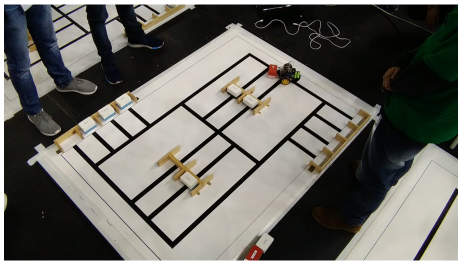


Figure 6.1: Photo taken during the competition at FNR 2019.

The problems that the team confronted in the competition are the errors already mentioned in Section 4.1.3. The robot's slippage and skid, especially, jeopardized the time dependency that the robot's control had. This was the reason that in round three the robot could not deliver the four boxes. In addition, as we increased the robot's speed after a successful run in a round, the probability of these errors occurring increased as well. The strategy was to guarantee all the boxes first, and after that, perform faster runs trying to achieve a better time.

6.2 Working Route

As can be seen in Figure 6.2 and already stressed in Section 5.1, the robot performs a similar path in relation to the spline generated by the control application with the waypoints inserted by the user.

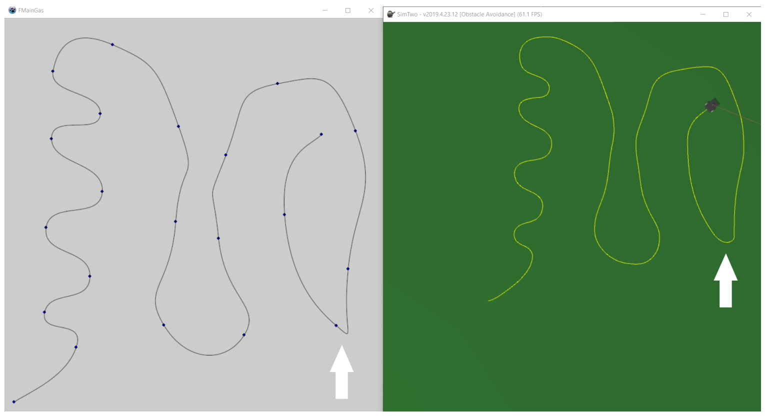


Figure 6.2: Application screenshot of the trajectory inserted by the user (left image) and the robot's path performed in simulation (right image).

Note the path similarity in Figure 6.2, the only time that the robot does not perform almost identical path is when it has abrupt curves, displayed by the white arrow. However, this is expected as the sampling causes this situation.

6.3 Path Planning Algorithms Comparison

At first, A* and RRT* were compared. The comparison approach will consider the longest processing time (as the path planning is recalculated every time a new obstacle is found), the execution time, and finally, the distance travelled. The first is simply the longer processing time for the algorithm to converge to the target point, which was, in this case study, the first time that the algorithm computed the trajectory. The second, is the time that the robot takes to reach the goal. The latter, will be the course taken by the robot during the test. It will be measured by odometry of the two wheel axes of the differential geometry. Note that the robot's speed will be the same to all the tests with both algorithms. The processing time will be measured in MATLAB where the algorithms codes run. For the execution time and distance traveled, both of them are measured in the simulator. Figure 6.3 illustrates the communication between SimTwo and MATLAB.

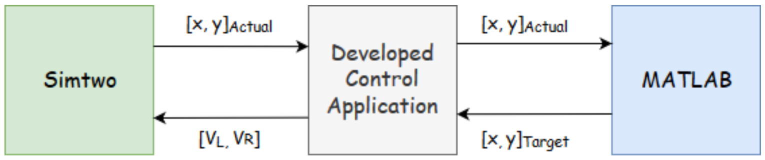


Figure 6.3: System Structure.

The time starts to count when the robot moves and only stops when it reaches the goal. The criteria to check if the robot reaches the goal, is a Euclidean distance between the actual robot's position and the goal position. If the distance magnitude is less than 0.1, then the code consider that the robot reached the goal.

In the same manner, the distance starts to be measured when the robot starts moving and stops when the reaches the goal. The simulated encoder has 360 pulses per revolution in each axis. In this sense, the distance traveled will be an arithmetic mean between the two axes, $Dist = \frac{(LeftAxis+RightAxis)}{2}$. Therefore, the distance performed in each axis is, $Axis = \frac{(2\times\pi\times r)}{360}$, which r is the wheel radius. Note that, $Dist$ will be measured in mm as the wheel radius is expressed in mm as shown in Section 4.1.

Several scenarios were tested for the algorithms. In this work, two scenarios are presented. They are similar, however with small adjustments that will change both of the algorithm optimal paths. In the first scenario, the robot will start in the middle of the map and its objective is to reach the red cuboid behind the several obstacles in the northeast part of Figure 6.4. It is clear that in Figure 6.4, the only possible path to the other side of the walls in by the narrow gap between the two walls in the southeast part. It is possible to assume that when the code starts to run, using a simulated LIDAR sensor described in Section 4.2, the robot sees only this gap. This happens because for these comparisons only presented in this Section, the LIDAR laser did not have a distance measurement limitation. Also, the area coverage was set for 360°.

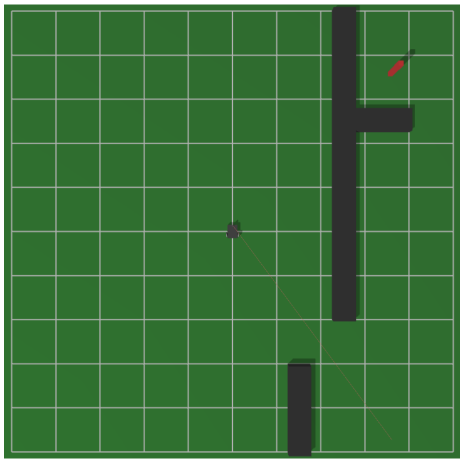


Figure 6.4: First scenario test in SimTwo Environment.

In the second scenario, small adjustments were made to force the algorithms to try out different approaches. As can be seen in Figure 6.5, when the simulation starts to run, the robot can not conclude that the gap in the southeast part of the map is the only possible path. In this way, it chooses the upper path because it is shorter than the lower path. The robot does not know if behind the upper walls in Figure 6.5 the path will be blocked. The explanation for each algorithm for why they choose the upper path are explained in their respective Sections 5.2.1 and 5.2.2.

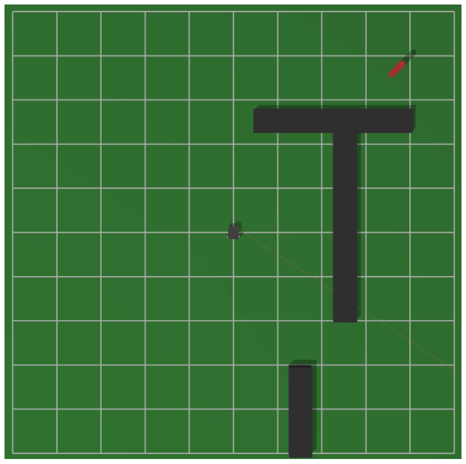


Figure 6.5: Second scenario test in SimTwo environment.

As RRT* algorithm is a stochastic process, ten tests were made for each scenario. This decision was made not only because of the non-deterministic behaviour of the RRT*, but also because in this way, computing the mean and standard deviation, the comparison will be more reliable.

For A* algorithm in each scenario, the optimal path will be the same in all tests. For this reason, only one test will be presented for each scenario. As for the non-deterministic behaviour of RRT*, results of two tests will be presented for each scenario for the sake of exemplification. All configurations were also applied for both algorithms. In other words, all the physical characteristics of the environment and the sensors were applied in SimTwo simulator to obtain the performances of A* and RRT*.

6.3.1 First Scenario

The first scenario deals with the movement of the robot starting from the center of the map, where the end point is placed in the upper right corner, represented by the red cuboid. The A* algorithm is expected to calculate the trajectory, bypassing the obstacles. Figure 6.6 illustrates the whole scenario with the trajectory found.

As already mentioned, with the trajectory created, the system sends the robot to reach the end point. Figure 6.7 shows the path traveled by the robot avoiding the obstacles created in the SimTwo environment. The entire analysis process in A* was repeated ten times assuming that for each test, the system was restarted. Then the data were collected and inserted in table format. Figure 6.8 reports the longest computation time, execution time and distance traveled that the robot spent to perform the route with A* algorithm.

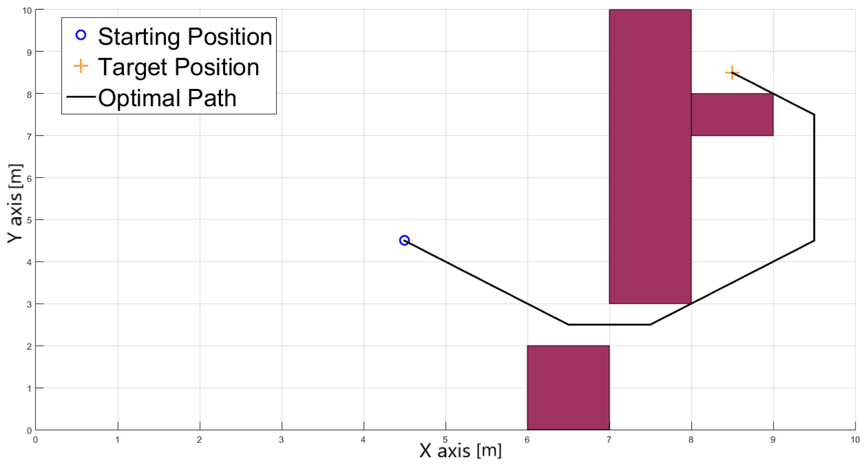


Figure 6.6: Illustration of path planned by A*.

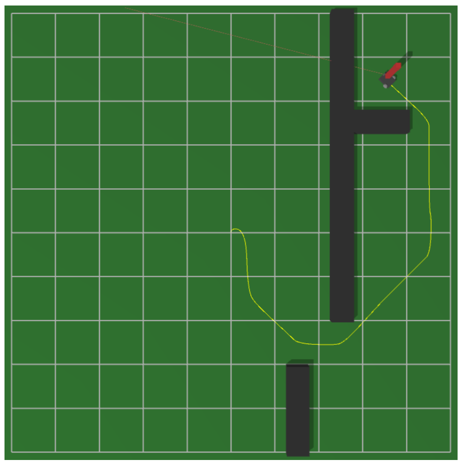


Figure 6.7: Path performed by the robot in the first scenario.

As can be noted, A* performs roughly the same results in all ten tests, performing similar runs in each test. The averages and standard deviations were computed and Table 6.2 show all the data obtained in this series of tests.

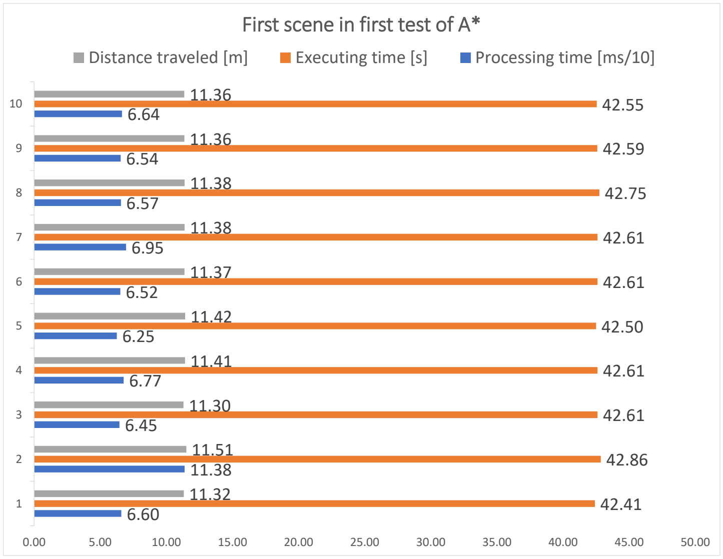


Figure 6.8: Processing time, executing time and distance traveled in first scenario of A*.

Table 6.2: First Scenario A* Results.

Averages		
Processing (s)	Executing (s)	Distance (m)
0.0706	42.6095	11.3805
Standard Deviations		
Processing (s)	Executing (s)	Distance (m)
0.0144	0.1183	0.0554

As in the previous series of tests, they were also applied for RRT* algorithm. Figure 6.9 shows the trajectory found to reach the target. After the trajectory found by the RRT*, the developed system can already send the information to the virtual robot. The environment of this approach can be seen in Figure 6.10.

All the analysis processed in RRT* was also repeated ten times considering that, for each test, the system is also restarted. The data from the ten tests are also in table format. Figure 6.11 shows the longest computation time, executing time and distance traveled that the robot took to perform the route with RRT* algorithm.

Differently from A*, RRT* presents susceptible differences in these tests. Computing

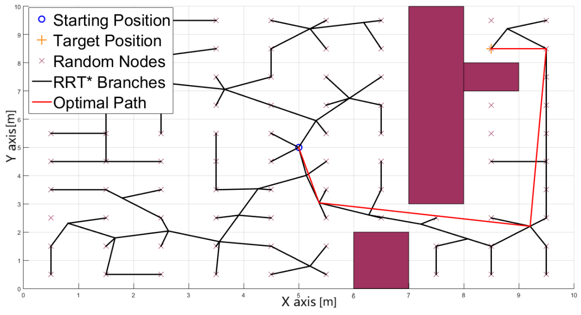


Figure 6.9: Illustration of path planned by RRT* in the first test.

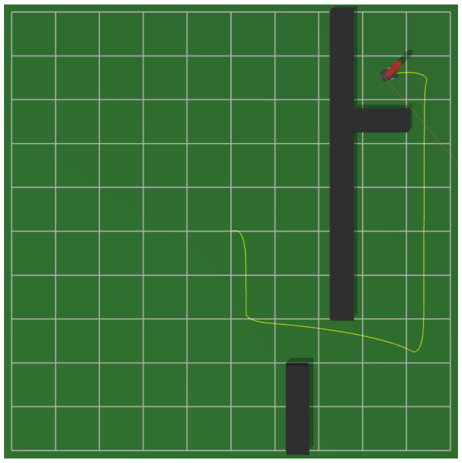


Figure 6.10: Path performed by robot in the first test.

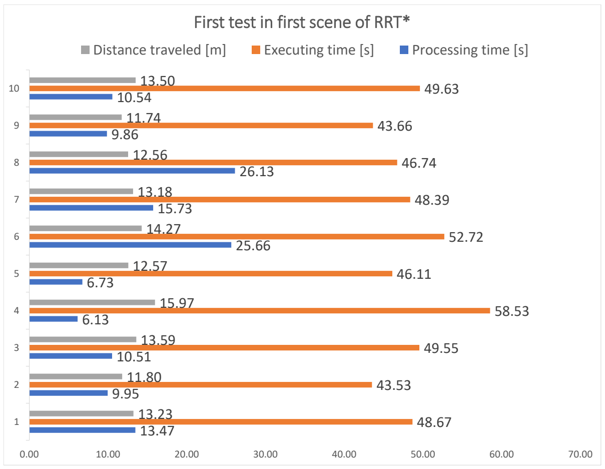


Figure 6.11: Processing time, executing time and distance traveled in first scenario of RRT*.

the average and standard deviation is possible to conclude that the path quality depends heavily with more processing time, i.e, more generated nodes. All these values are demonstrated in Table 6.3.

Table 6.3: First Scenario RRT* Results.

Averages		
Processing (s)	Executing (s)	Distance (m)
13.4712	48.7516	13.2409
Standard Deviations		
Processing (s)	Executing (s)	Distance (m)
6.7515	4.2092	1.1815

6.3.2 Second Scenario

For the analysis of the second scenario, the initial point of the robot is also the center of the map, and the target point is in the upper right corner. However, in this analysis, one of the obstacles has been opened up. In this way, it is expected that the algorithms have performances different from those obtained in the first scenario. Figure 6.12 shows the route found by A* for this scenario, and the movements of the virtual robot in SimTwo can be seen in Figure 6.13.

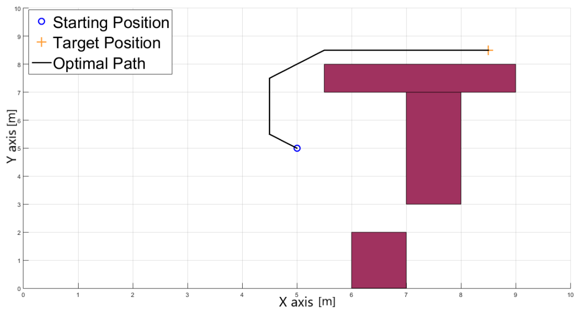


Figure 6.12: Illustration of path planned by A* in the second scenario.

The paths generated by A* in Figure 6.6 and Figure 6.12 were different, and this is expected due to the complexity differences between the scenarios. As in the first scenario, the tests were also repeated ten times and for each time the system was also restarted.

The data of all of these tests are provided in table format. The longest processing time, execution time and distance traveled are shown in Figure 6.14.

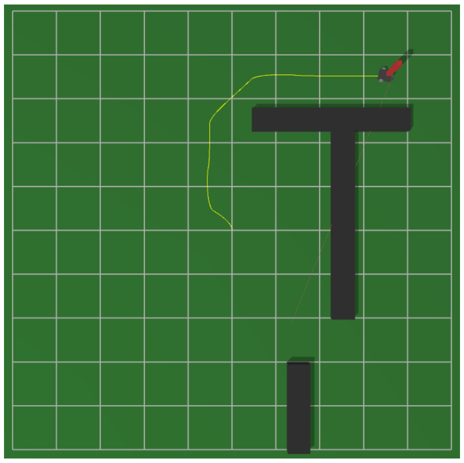


Figure 6.13: Path performed by robot in the second scenario.

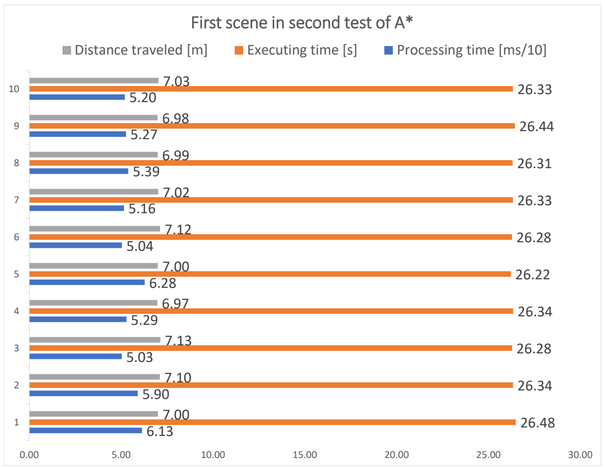


Figure 6.14: Processing time, executing time and distance traveled in second scenario of A*.

The averages and standard deviations are lower than the first scenario. This is expected since this scenario is less complex. The data are summarized in Table 6.4.

Table 6.4: Second Scenario A* Results.

Averages		
Processing (s)	Executing (s)	Distance (m)
0.0546	26.3355	7.0354
Standard Deviations		
Processing (s)	Executing (s)	Distance (m)
0.0043	0.0724	0.0585

In this second scenario, the tests are also carried out for RRT* performance analysis. Figure 6.15 indicates the route calculated by the RRT* algorithm. The virtual robot movement, in the simulated environment, is shown in Figure 6.16.

All the paths are similar as the distance to reach the goal is considerably lower than scenario one, and because of that, the algorithm converged much more faster. Figure 6.17 shows the longest processing time, execution time and distance traveled.

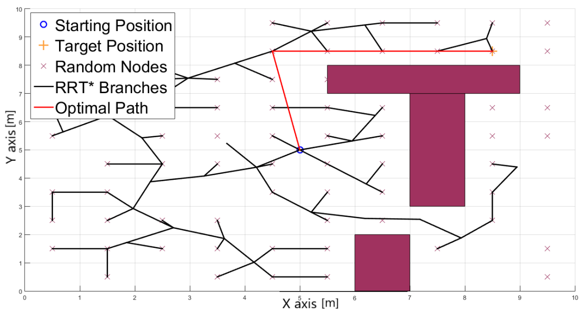


Figure 6.15: Illustration of path planned by RRT* in the second scenario during test one.

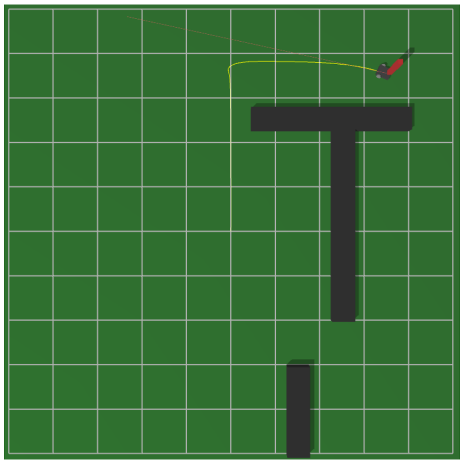


Figure 6.16: Path performed by robot in the second scenario during test one.

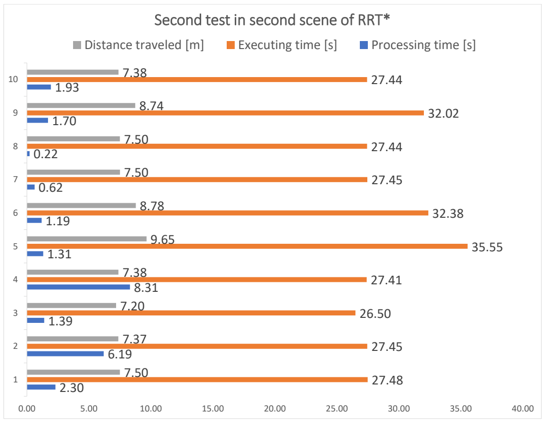


Figure 6.17: Processing time, executing time and distance traveled in second scenario of RRT*.

The averages and standard deviations are measured, and Table 6.5 reports them. Again, the values are lower due to the complexity of the scenario.

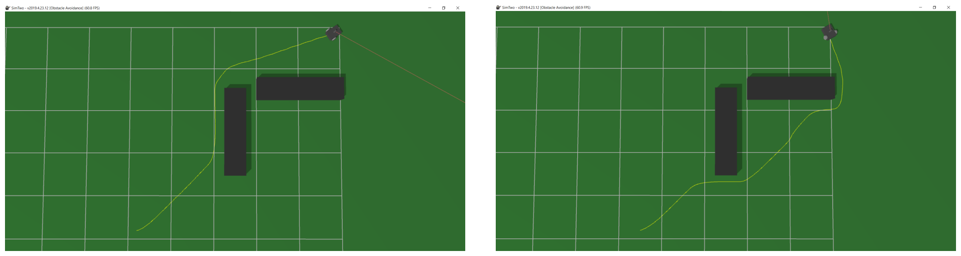
Table 6.5: Second Scenario RRT* Results.

Averages		
Processing (s)	Executing (s)	Distance (m)
2.5156	29.111	7.8997
Standard Deviations		
Processing (s)	Executing (s)	Distance (m)
2.4799	2.8979	0.7949

As can be noted, running the path planning in MATLAB with communication is not efficient. Thus, an A* variation was used, Binary Heap A*, available at [99]. This variation, provides better performance by using binary heap to store and access data in memory. In addition, by the fact of being run in the control application, the communication delay was removed. Moreover, this algorithm uses a proportionality constant to generate sub optimal paths but with shorter processing times [100]. Therefore, it was chosen to use with the gas search algorithm. This decision provided much better performances as this work requires a real time obstacle avoidance system.

6.4 Obstacle Avoidance

Two scenarios were tested to prove the Binary Heap A* path planning algorithm developed for this work. In the first scenario, the robot is inserted near the origin $[x, y] = [0.2, 0.2]$ and set to go to the point $[x, y] = [5, 5]$. There are two obstacles blocking the direct path to the target. Figure 6.18a displays the first scenario.



(a) First position. (b) Second position.

Figure 6.18: Path planning test scenario using a Binary Heap A*.

In another test, the robot was inserted a little bit to the right $[x, y] = [0.5, 0.2]$, to force the robot to choose the right path, as the heuristic cost to reach the target point is lower. Note that, as explained in Section 4.2.1, the LIDAR model works from 3 cm to 400 cm. In this concept, the robot can not see obstacles beyond this range. Thus, the robot continues the right path until it sees the rightmost obstacle. However, when it sees this obstacle, the heuristic cost to go back to the left path is still higher. Thus, the robot continues with the first path choice. Figure 6.18b presents this behaviour.

Finally, in the second scenario, two more obstacles are placed to trick the path planning. As can be seen in Figure 6.19, the robot is entered in $[x, y] = [1.5, 0.2]$ to decrease even further the heuristic cost to the robot perform the right path. However, as the robot can not see beyond 400 cm, the robot does not know the obstacles until they reach its range. In this idea, the robot keeps going to the farthest path even though the left one was the shortest. Making the robot circle all the obstacles.

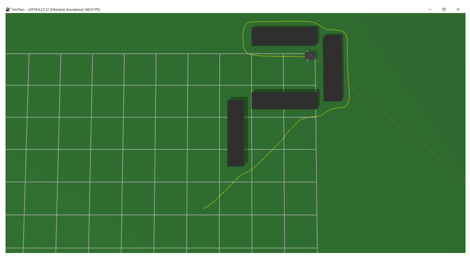
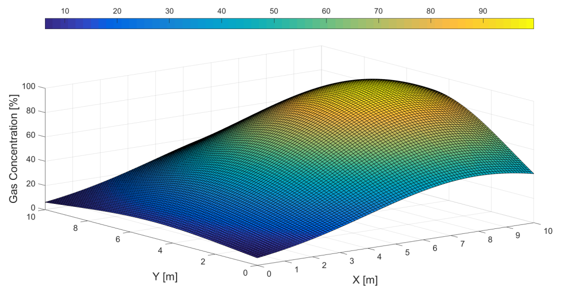


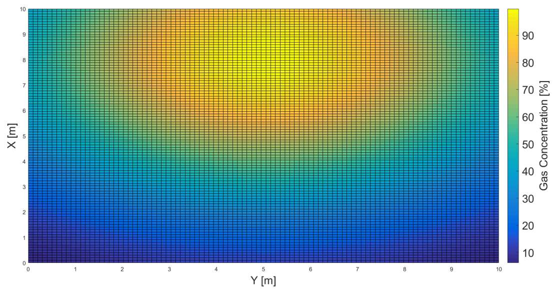
Figure 6.19: Path planning test second scenario using a Binary Heap A*.

6.4.1 Gas search algorithm without obstacles

As explained in Section 4.3, the algorithm does not takes assumption on the gas model and its maximum concentration value, as in the real scenario this can not be predicted. In this way, the algorithm is tested without obstacles in just one scenario, because the main objective of this work is to validate the algorithm with the obstacle avoidance feature. The gas position model in this test can be seen in Figure 6.20.



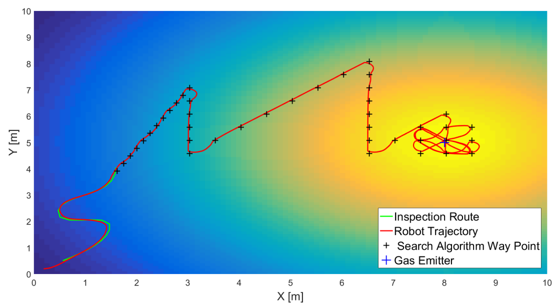
(a) Isometric view.



(b) Top view.

Figure 6.20: Gas model in the first scenario in the combined test.

The gas concentration is color coded as can be noted in Figure 6.20. This was done to ease the understanding of the algorithm behaviour. Moreover, Figure 6.21a illustrates the comparison between the path performed by the robot and the route made by the user in the application as well as the gas search algorithm points.



(a) Comparison of the working route and gas search algorithm.

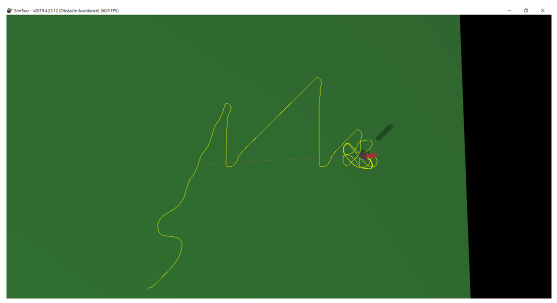
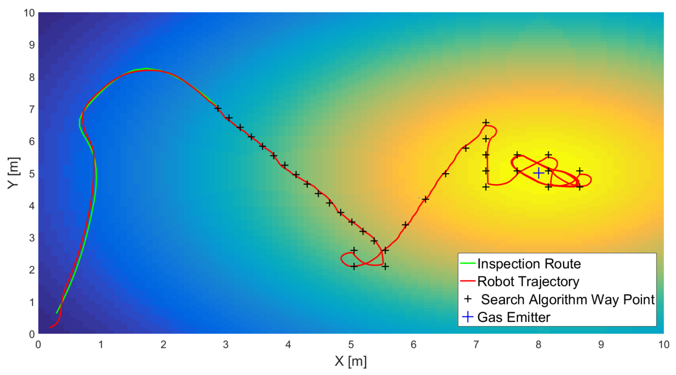

(b) Path planning created in SimTwo.

Figure 6.21: Path planning test along with gas search algorithm without obstacles.

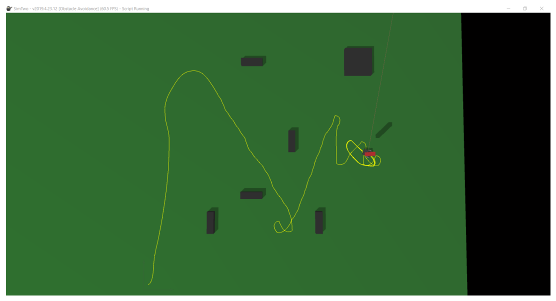
Figure 6.21b presents the simulation result that was represented by Figure 6.21a. Note that, when the robot abandons the working route and starts to search for the gas leak, the black plus symbols in Figure 6.21a begin to appear on the plot (which are the points computed by the gas search algorithm). In other words, the robot starts to follow these points. In addition, as explained in Section 4.2, the search algorithm makes the robot keep going in the same direction until the gradient becomes negative, then it changes its direction to a new positive gradient. Finally, when it reaches the maximum point, it keeps circling the peak value. It was decided to not implement a stop point as the idea behind this work is to find the gas leak, and let the robot circle it.

6.4.2 Gas search with obstacle avoidance

Finally, two challenging scenarios were tested with the gas algorithm combined with the path planning. The robot started near the environment origin. In addition, the gas model was placed in $[x, y] = [8, 5]$, and its surface plot can be seen in Figure 6.20. The first analysis is applied to obstacles in the same gas model in the previous verification. In this concept, the working route is defined (green line in Figure 6.22a) to the robot start the test. Figure 6.22 displays the robot and the algorithm behaviour.



(a) Working route, path performed by the robot during the test.

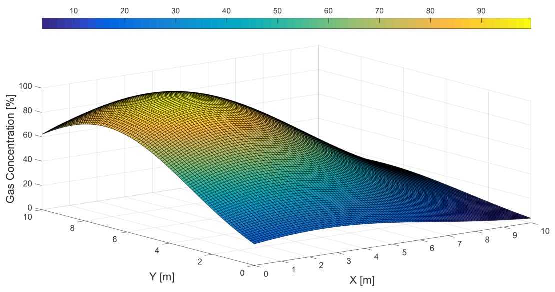


(b) Gas search algorithm with obstacles in SimTwo.

Figure 6.22: Path planning test along with gas search algorithm with obstacles.

As can be seen in Figures 6.22a and 6.22b, the robot could perform the search with several obstacles placed in the scenario. Moreover, note that the robot followed the gas search points smoothly (black plus symbols in Figure 6.22a). Additionally, when the gas search algorithm sent a point that the robot could not reach, the control application with the path planning, sent the robot to the nearest point accessible. The robot with the algorithms converged to the gas leakage source without any problems.

In the second scenario the gas model position is changed to $[x, y] = [2.5, 7]$ and the robot to $[x, y] = [7.5, 0.5]$. Figure 6.23 displays the gas model in the second scenario.


(a) Isometric view.

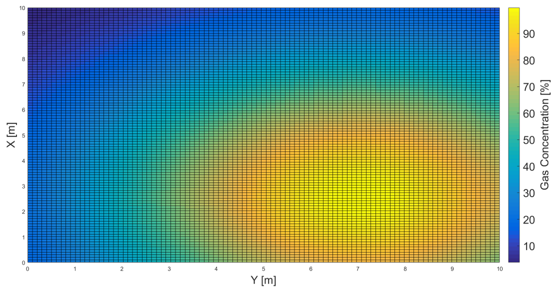
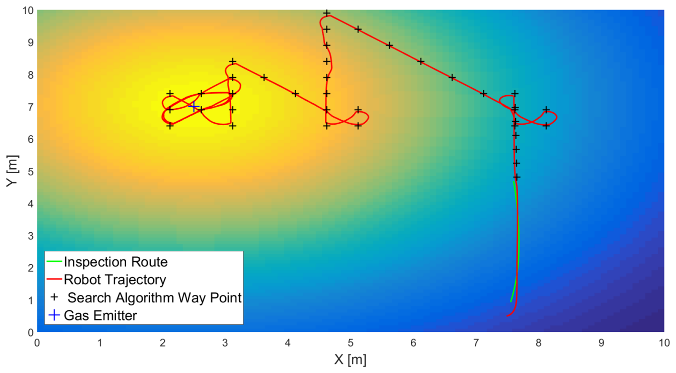

(b) Top view.

Figure 6.23: Gas model in the second scenario in the combined test.

In addition, Figure 6.24 displays the robot's path behaviour in the second scenario with the obstacles.



(a) Working route, path performed by the robot during second test.

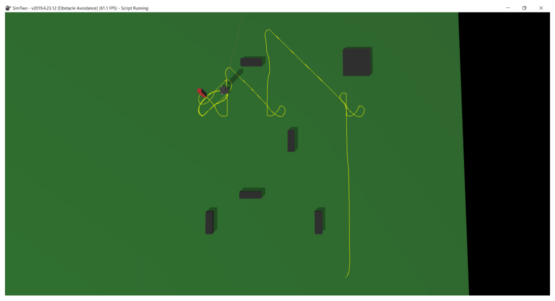

(b) Gas search algorithm with obstacles in Simtwo, during the second test.

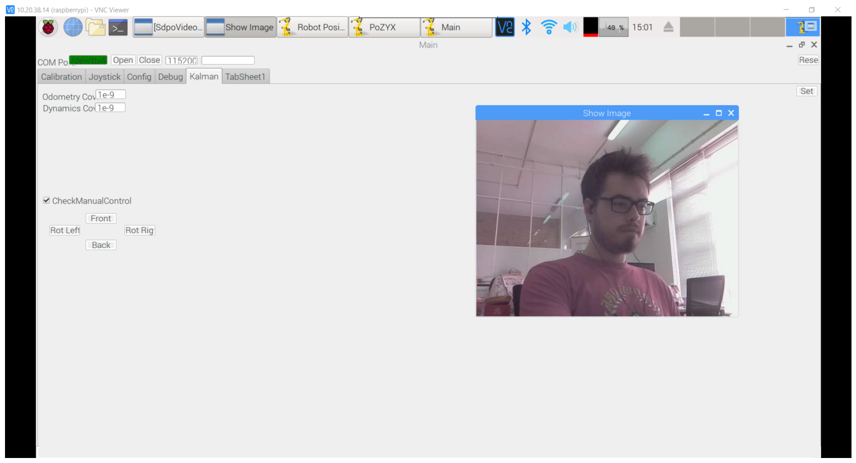
Figure 6.24: Path planning test along with gas search algorithm with obstacles, during the second test.

As can be noted in Figure 6.24, the robot could perform the search without colliding with any obstacles. The gas search algorithm behaviour is well seen in directions change in the robot's path represented by the yellow line in Figures 6.21b, 6.22b and 6.24b. As explained in Section 4.3, the robot keeps moving in the direction that has a positive gradient and, when it becomes negative, it changes the direction.

Again, the green line, which is the working route and the red line, which is the path performed by the robot, both of them are similar (seen in Figures 6.21a, 6.22a and 6.24a). This displays that the robot's controller is well adjusted. Moreover, soon after the robot quits the working route (when the black plus symbols starts to appear), the robot keeps going upwards in the Y axis until the gradient becomes negative and then changes its direction. This happens two more times, until the robot's keeps circling the gas source.

6.5 Remote Operation

As mentioned in Chapter 1, if needed, the robot could be remotely operated by the computer. The robot's control by buttons, in the robot's software, made by L. Piardi [3] and seen in Figure 6.25, was adapted to work with the computer keyboard and integrated with camera feedback. For the camera visualization, it was used "5dpo Component Library for Lazarus" made by the 5dpo Robotic Soccer Team found in [101]. Thus, the robot can be remotely operated via Wi-Fi transmission with visual feedback. Figure 6.25 displays the feature.



115200

Odometry Cov 1e-9

Dynamics Co 1e-9

☑ CheckManualControl

Figure 6.25: Robot's control with visual feedback by camera.

6.6 Real Test

A real test was made with the real robot. To setup the test area, the four fixed anchors were placed to form a 2×2 m inertial basis. The idea was to made the robot to follow the working route generated by the waypoints selected by the user, gathering the gas sensors data during the route. As already mentioned (see Section 2.4.3), the robot's localization system was already validated in [8]. This way, there was no concern regarding the robot's pose reliability. Therefore, Figure 6.26 displays the robot's path result.

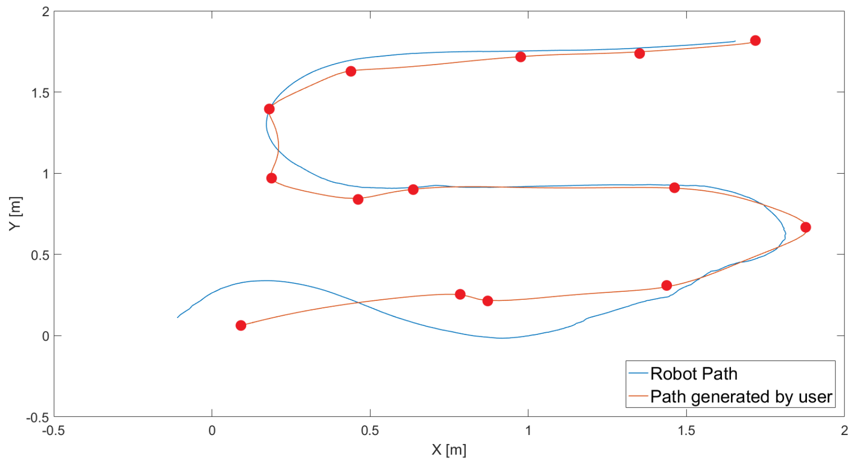


Figure 6.26: Working route generated by the user and the robot's performed path. Red circles are the waypoints inserted by the user.

This result is expected to happen as the UWB system has precision in the range of 10 to 20 cm in locating the movable tag. In addition, summed with the noise, uncertainty and errors in dead reckoning (see Section 4.1.3), the robot could follow the path with little struggle. Note that, this space for testing was small (4 m^2), and because of the robot size (see Table 4.1), the robot had the struggle in the beginning of the motion to adjust accordingly to the path. In addition, if the test space was bigger (which is the case for industries), this error would be less relevant.

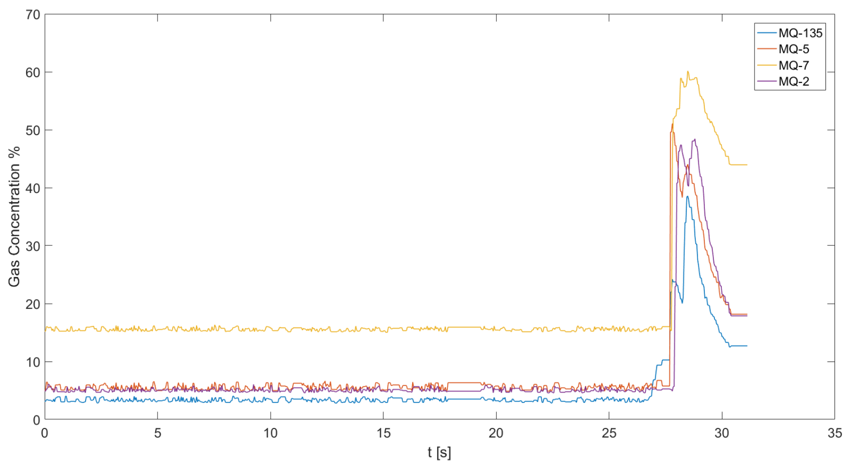


Figure 6.27: Gas sensors data during the working route. Time does not correspond to the working route time.

Figure 6.27 shows the gas sensors during the working route. When the robot finished the route, a lighter was lighted slightly above the sensors just to show their reaction. Additionally, the butane inside the lighter was expelled. Naturally, as explained in 4.1.2, this sensors are not built to distinct the gases and they detect several types of gases. Therefore, as they were exposed to carbon dioxide, carbon monoxide and butane, every sensor had a peak in its analog value. Note that as the Figure 6.27 legend displays, this time does not correspond the exact time that the robot took to reach the final way point. This happened, because the lighter was manually ignited before stopping the system.

6.7 Results summary

During the course of this work, several results have been made. A control application was implemented that works as a human-machine interface. This app generate a working route for the AGV using splines. The path is sampled, and with the robot's control that was developed, the robot follows the trajectory faithfully in simulation. During the real test, the path performed was similar and inside the localization system error range. Although the result was similar for a simple controller and for a small test space for a big robot, it can be improved. Three path plannings were developed and their performance compared. A gas search algorithm was developed. The Binary Heap A* path planning was combined with the LIDAR sensor to develop an obstacle avoidance system. The gas search algorithm was integrated with the obstacle avoidance system and the working route to search for a gas source autonomously. The robot's control was integrated with keyboard to be remotely operated with visual feedback by camera. Finally, a low cost AGV was assembled and a manual was written to encourage other students to compete (see Appendix D). A team was formed to compete in the FNR 2019 - Gondomar with the AGV and we returned victorious (see A and Figure 3.1).

Chapter 7

Conclusions and Future Works

7.1 Developed Works

During the course of this work, several developments were made. First, the low cost AGV was assembled and its assembly manual was made to assist the competition, see Appendix D. With the help of the rest of the team, IPB@Factory managed to win the Robot@Factory Lite Competition. This covered the working route scope of this work. In addition, a control application was made for the AMR that contains all the mobile robot features. These features are: the working route generated by the user by inserting waypoints, the point follower P controller, obstacle avoidance algorithm with the aid of the LIDAR data which is gathered and processed by trigonometrical and mathematical computations (dilation), three path plannings were developed and their performance compared, a gradient based gas search algorithm which does not take in assumption the concentration peak value and the gas model to function, the mobile robot hardware adaptation (gas sensors, Arduino Nano and the circuit necessary to drive, control and communicate between them and the Raspberry Pi), the FIR low pass filter that process the gas sensors data, and finally, the application that can operate the robot remotely by the computer using the keyboard and the Raspberry Pi camera. Thus, all the objectives first proposed in the beginning of this work were overcome.

7.2 Future Works

This work covers several areas of science and has much potential to develop, given the time and dedication. The topics that could be developed:

- Embed motors with encoders in the low cost AGV to mitigate/remove the non-systematic errors, by using odometry, that the robot has. Moreover, two AGVs can be used to compete, programming them to work together and complete the trial in much less time. Additionally, a better controller can be implemented, for instance, PID. Thus, elevating the competitiveness of the Robot@Factory Lite competition.
- Develop a better controller for the mobile robot. Such as a fuzzy or PID to enhance the robot's ability to follow the path.
- Real tests with the obstacle avoidance algorithm to test the LIDAR device with the robot control.
- The gas algorithm search can be improved to converge to the point faster and to consider the wind in the gas distribution.
- Better path plannings can be implemented and tested.
- A joystick can developed to remotely operate the robot, instead of using the key-board.

Bibliography

[1] S. R. Systems, *Gas detector – air quality monitoring robot*, Last accessed 27 March 2019. [Online]. Available: `https://smprobotics.com/products_autonomous_ugv/area-and-perimeter-gas-monitoring-robot/`.

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

[3] L. Piardi, "Application of a mobile robot to spatial mapping of radioactive substances in indoor environment," Master's thesis, Instituto Politécnico de Bragança, 2018.

[4] R. R. Murphy, "Trial by fire [rescue robots]," *IEEE Robotics & Automation Magazine*, vol. 11, no. 3, pp. 50–61, 2004.

[5] D. Voth, "A new generation of military robots," *IEEE Intelligent Systems*, vol. 19, no. 4, pp. 2–3, 2004.

[6] G. Dogangil, B. Davies, and F. Rodriguez y Baena, "A review of medical robotics for minimally invasive soft tissue surgery," *Proceedings of the Institution of Mechanical Engineers, Part H: Journal of Engineering in Medicine*, vol. 224, no. 5, pp. 653–679, 2010.

[7] J. Lima and P. Costa, "Ultra-wideband time of flight based localization system and odometry fusion for a scanning 3 dof magnetic field autonomous robot," in *Iberian Robotics conference*, Springer, 2017, pp. 879–890.

[8] 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," in *Proceedings of the 15th International Conference on Informatics in Control, Automation and Robotics - Volume 2: ICINCO,*, INSTICC, SciTePress, 2018, pp. 341–348, ISBN: 978-989-758-321-6. DOI: `10.5220/0006862203510358`.

[9] H. Fazlollahtabar and M. Saidi-Mehrabad, *Autonomous Guided Vehicles: Methods and Models for Optimal Path Planning*, ser. Studies in Systems, Decision and Control. Springer International Publishing, 2015, ISBN: 9783319147475. [Online]. Available: `https://books.google.pt/books?id=Q05WBgAAQBAJ`.

[10] S. Automation, *The history of agvs*, Last accessed 22 May 2019. [Online]. Available: `http://www.agvsystems.com/history-agvs/`.

[11] C. Russell, *Agv navigation pros and cons*, Last accessed 24 May 2019. [Online]. Available: `https://www.scottautomation.com/assets/AGV-Navigation-Pros-Cons-Presentation.pdf`.

[12] D. Indelicato, "Development of an algorithm to find loads on the floor relatively to the position of the robot," PhD thesis, Politecnico di Torino, 2018.

[13] G. KG, *Introduction inductive track guidance*, Last accessed 20 July 2019. [Online]. Available: `https://www.goetting-agv.com/components/inductive/introduction`.

[14] Transbotics, *Navigation technology*, Last accessed 24 May 2019. [Online]. Available: `https://www.transbotics.com/learning-center/guidance-navigation`.

[15] S. Automation, *Tape-free, target-free internal agv / agc navigation*, Last accessed 24 May 2019. [Online]. Available: `https://www.agvsystems.com/navigation-landing/`.

[16] B. Solutions, *Optimize your warehouse with vision-guided technology*, Last accessed 24 May 2019. [Online]. Available: `https://www.bastiansolutions.com/solutions/technology/automated-guided-vehicles/vision-guided-vehicles/`.

[17] BALYO, *Geoguidance : Navigation without infrastructure.* Last accessed 24 May 2019. [Online]. Available: `https://www.balyo.com/Technology/Navigation`.

[18] Transbotics, *Automatic guided vehicles (agv) drive & steering options*, Last accessed 24 May 2019. [Online]. Available: `https://www.transbotics.com/learning-center/drive-steering`.

[19] L. Eitel, *Electric actuators for better robotic-vehicle steering*, Last accessed 20 July 2019. [Online]. Available: `https://www.therobotreport.com/electric-actuators-for-better-robotic-vehicle-steering/`.

[20] S. Automation, *Agv basics*, Last accessed 24 May 2019. [Online]. Available: `http://www.agvsystems.com/agvs-basics/basics-agvs/`.

[21] 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.

[22] C. E. Miller, A. W. Tucker, and R. A. Zemlin, "Integer programming formulation of traveling salesman problems," *Journal of the ACM (JACM)*, vol. 7, no. 4, pp. 326–329, 1960.

[23] J. T. Schwartz and M. Sharir, "On the "piano movers" problem. ii. general techniques for computing topological properties of real algebraic manifolds," *Advances in applied Mathematics*, vol. 4, no. 3, pp. 298–351, 1983.

[24] J. H. Reif, "Complexity of the generalized mover's problem.," HARVARD UNIV CAMBRIDGE MA AIKEN COMPUTATION LAB, Tech. Rep., 1985.

[25] Y. Huang, Z. Cao, and E. Hall, "Region filling operations for mobile robot using computer graphics," in *Proceedings. 1986 IEEE International Conference on Robotics and Automation*, IEEE, vol. 3, 1986, pp. 1607–1614.

[26] B. Siciliano, "Kinematic control of redundant robot manipulators: A tutorial," *Journal of intelligent and robotic systems*, vol. 3, no. 3, pp. 201–212, 1990.

[27] J. Barraquand and J.-C. Latombe, "Robot motion planning: A distributed representation approach," *The International Journal of Robotics Research*, vol. 10, no. 6, pp. 628–649, 1991.

[28] E. Ralli and G. Hirzinger, "Fast path planning for robot manipulators using numerical potential fields in the configuration space," in *Proceedings of IEEE/RSJ International Conference on Intelligent Robots and Systems (IROS'94)*, IEEE, vol. 3, 1994, pp. 1922–1929.

[29] M. Ollis and A. Stentz, "First results in vision-based crop line tracking," in *Proceedings of IEEE International Conference on Robotics and Automation*, IEEE, vol. 1, 1996, pp. 951–956.

[30] S. T. A. W. DieterFox and T. D. T. M. TimoSchmidt, "Map learning and high-speed navigation in rhino," 1998.

[31] C. Pradalier, J. Hermosillo, C. Koike, C. Braillon, P. Bessiere, and C. Laugier, "Safe and autonomous navigation for a car-like robot among pedestrian," in *IARP Int. Workshop on Service, Assistive and Personal Robots*, 2003.

[32] J.-H. Hwang, R. C. Arkin, and D.-S. Kwon, "Mobile robots at your fingertip: Bezier curve on-line trajectory generation for supervisory control," in *Proceedings 2003 IEEE/RSJ International Conference on Intelligent Robots and Systems (IROS 2003)(Cat. No. 03CH37453)*, IEEE, vol. 2, 2003, pp. 1444–1449.

[33] K. R. Simba, N. Uchiyama, and S. Sano, "Real-time trajectory generation for mobile robots in a corridor-like space using bézier curves," in *Proceedings of the*

2013 IEEE/SICE International Symposium on System Integration, IEEE, 2013, pp. 37–41.

[34] X. Yang, Z. Zeng, J. Xiao, and Z. Zheng, "Trajectory planning for robocup msl mobile robots based on bézier curve and voronoi diagram," in *2015 IEEE International Conference on Information and Automation*, IEEE, 2015, pp. 2552–2557.

[35] J. Borenstein and Y. Koren, "The vector field histogram-fast obstacle avoidance for mobile robots," *IEEE transactions on robotics and automation*, vol. 7, no. 3, pp. 278–288, 1991.

[36] I. Ulrich and J. Borenstein, "Vfh+: Reliable obstacle avoidance for fast mobile robots," in *Proceedings. 1998 IEEE international conference on robotics and automation (Cat. No. 98CH36146)*, IEEE, vol. 2, 1998, pp. 1572–1577.

[37] ——, "Vfh/sup*: Local obstacle avoidance with look-ahead verification," in *Proceedings 2000 ICRA. Millennium Conference. IEEE International Conference on Robotics and Automation. Symposia Proceedings (Cat. No. 00CH37065)*, IEEE, vol. 3, 2000, pp. 2505–2511.

[38] L. Kavraki, P. Svestka, and M. H. Overmars, *Probabilistic roadmaps for path planning in high-dimensional configuration spaces*. Unknown Publisher, 1994, vol. 1994.

[39] S. A. Wilmarth, N. M. Amato, and P. F. Stiller, "Maprm: A probabilistic roadmap planner with sampling on the medial axis of the free space," in *ICRA*, 1999, pp. 1024–1031.

[40] T. Siméon, J.-P. Laumond, and C. Nissoux, "Visibility-based probabilistic roadmaps for motion planning," *Advanced Robotics*, vol. 14, no. 6, pp. 477–493, 2000.

[41] R. Bohlin and L. E. Kavraki, "Path planning using lazy prm," in *Proceedings 2000 ICRA. Millennium Conference. IEEE International Conference on Robotics and Automation. Symposia Proceedings (Cat. No. 00CH37065)*, IEEE, vol. 1, 2000, pp. 521–528.

[42] E. Plaku, K. E. Bekris, B. Y. Chen, A. M. Ladd, and L. E. Kavraki, "Sampling-based roadmap of trees for parallel motion planning," *IEEE Transactions on Robotics*, vol. 21, no. 4, pp. 597–608, 2005.

[43] S. Lahouar, S. Zeghloul, and L. Romdhane, "Real-time path planning for multi-dof manipulators in dynamic environment," *International Journal of Advanced Robotic Systems*, vol. 3, no. 2, p. 20, 2006.

[44] J. Yao, C. Lin, X. Xie, A. J. Wang, and C.-C. Hung, "Path planning for virtual human motion using improved a* star algorithm," in *2010 Seventh international conference on information technology: new generations*, IEEE, 2010, pp. 1154–1158.

[45] W. Y. Loong, L. Z. Long, and L. C. Hun, "A star path following mobile robot," in *2011 4th International conference on mechatronics (ICOM)*, IEEE, 2011, pp. 1–7.

[46] F. Duchoň, A. Babinec, M. Kajan, P. Beňo, M. Florek, T. Fico, and L. Jurišica, "Path planning with modified a star algorithm for a mobile robot," *Procedia Engineering*, vol. 96, pp. 59–69, 2014.

[47] P. Sudhakara and V. Ganapathy, "Trajectory planning of a mobile robot using enhanced a-star algorithm," *Indian Journal of Science and Technology*, vol. 9, no. 41, pp. 1–10, 2016.

[48] Y. Cheng and G. Y. Wang, "Mobile robot navigation based on lidar," in *2018 Chinese Control And Decision Conference (CCDC)*, IEEE, 2018, pp. 1243–1246.

[49] C.-b. Moon and W. Chung, "Kinodynamic planner dual-tree rrt (dt-rrt) for two-wheeled mobile robots using the rapidly exploring random tree," *IEEE Transactions on industrial electronics*, vol. 62, no. 2, pp. 1080–1090, 2015.

[50] L. Palmieri and K. O. Arras, "A novel rrt extend function for efficient and smooth mobile robot motion planning," in *2014 IEEE/RSJ International Conference on Intelligent Robots and Systems*, IEEE, 2014, pp. 205–211.

[51] C. Ma, Y. Zhang, Q. Zhao, and K. Bai, "6r serial manipulator space path planning based on rrt," in *2016 8th International Conference on Intelligent Human-Machine Systems and Cybernetics (IHMSC)*, IEEE, vol. 2, 2016, pp. 99–102.

[52] T. Brito, J. Lima, P. Costa, and L. Piardi, "Dynamic collision avoidance system for a manipulator based on rgb-d data," in *Iberian Robotics conference*, Springer, 2017, pp. 643–654.

[53] D.-Q. He, H.-B. Wang, and P.-F. Li, "Robot path planning using improved rapidly-exploring random tree algorithm," in *2018 IEEE Industrial Cyber-Physical Systems (ICPS)*, IEEE, 2018, pp. 181–186.

[54] M. Lauer, S. Lange, and M. Riedmiller, "Calculating the perfect match: An efficient and accurate approach for robot self-localization," in *Robot Soccer World Cup*, Springer, 2005, pp. 142–153.

[55] J. Zhang, "Designing a cost-effective and reliable pipeline leak-detection system," *Pipes and Pipelines International*, vol. 42, no. 1, pp. 20–26, 1997.

[56] N. Turner *et al.*, "Hardware and software techniques for pipeline integrity and leak detection monitoring," in *Offshore Europe*, Society of Petroleum Engineers, 1991.

[57] O. Hunaidi, W. Chu, A. Wang, and W. Guan, "Detecting leaks in plastic pipes," *Journal-American Water Works Association*, vol. 92, no. 2, pp. 82–94, 2000.

[58] A. Kroll, W. Baetz, and D. Peretzki, "On autonomous detection of pressured air and gas leaks using passive ir-thermography for mobile robot application," in *Robotics and Automation, 2009. ICRA'09. IEEE International Conference on*, IEEE, 2009, pp. 921–926.

[59] W. Baetz, A. Kroll, and G. Bonow, "Mobile robots with active ir-optical sensing for remote gas detection and source localization," in *Robotics and Automation, 2009. ICRA'09. IEEE International Conference on*, IEEE, 2009, pp. 2773–2778.

[60] R. A. Russell, D. Thiel, R. Deveza, and A. Mackay-Sim, “A robotic system to locate hazardous chemical leaks,” in *Robotics and Automation, 1995. Proceedings., 1995 IEEE International Conference on*, IEEE, vol. 1, 1995, pp. 556–561.

[61] H. Ishida, H. Tanaka, H. Taniguchi, and T. Moriizumi, “Mobile robot navigation using vision and olfaction to search for a gas/odor source,” *Autonomous Robots*, vol. 20, no. 3, pp. 231–238, 2006.

[62] D. Martinez, O. Rochel, and E. Hugues, “A biomimetic robot for tracking specific odors in turbulent plumes,” *Autonomous Robots*, vol. 20, no. 3, pp. 185–195, 2006.

[63] C. Lytridis, E. E. Kadar, and G. S. Virk, “A systematic approach to the problem of odour source localisation,” *Autonomous Robots*, vol. 20, no. 3, pp. 261–276, 2006.

[64] A. Loutfi and S. Coradeschi, “Smell, think and act: A cognitive robot discriminating odours,” *Autonomous Robots*, vol. 20, no. 3, pp. 239–249, 2006.

[65] S. Larionova, N. Almeida, L. Marques, and A. T. de Almeida, “Olfactory coordinated area coverage,” *Autonomous Robots*, vol. 20, no. 3, pp. 251–260, 2006.

[66] T. Lochmatter, X. Raemy, L. Matthey, S. Indra, and A. Martinoli, “A comparison of casting and spiraling algorithms for odor source localization in laminar flow,” in *Robotics and Automation, 2008. ICRA 2008. IEEE International Conference on*, IEEE, 2008, pp. 1138–1143.

[67] M. Wandel, A. J. Lilienthal, T. Duckett, U. Weimar, and A. Zell, “Gas distribution in unventilated indoor environments inspected by a mobile robot,” in *IEEE international conference on advanced robotics, ICAR 2003, Coimbra, Portugal, June 30-July 3, 2003*, University of Coimbra, vol. 1, 2003, pp. 507–512.

[68] G. Ferri, E. Caselli, V. Mattoli, A. Mondini, B. Mazzolai, and P. Dario, “A biologically-inspired algorithm for gas/odor source localization in an indoor environment with no strong airflow: First experimental results,” in *Proceedings of IEEE International Conference on Robotics and Automation*, 2007, pp. 566–571.

[69] L. Marques, A. Martins, and A. T. de Almeida, "Environmental monitoring with mobile robots," in *2005 IEEE/RSJ International Conference on Intelligent Robots and Systems*, IEEE, 2005, pp. 3624–3629.

[70] G. Kowadlo and R. A. Russell, "Robot odor localization: A taxonomy and survey," *The International Journal of Robotics Research*, vol. 27, no. 8, pp. 869–894, 2008.

[71] M. Wandel, A. J. Lilienthal, T. Duckett, U. Weimar, and A. Zell, "Gas distribution in unventilated indoor environments inspected by a mobile robot," in *IEEE international conference on advanced robotics, ICAR 2003, Coimbra, Portugal, June 30-July 3, 2003*, University of Coimbra, vol. 1, 2003, pp. 507–512.

[72] L. Marques, A. Martins, and A. T. de Almeida, "Environmental monitoring with mobile robots," in *2005 IEEE/RSJ International Conference on Intelligent Robots and Systems*, IEEE, 2005, pp. 3624–3629.

[73] M. Trincavelli, "Gas discrimination for mobile robots," *KI-Künstliche Intelligenz*, vol. 25, no. 4, pp. 351–354, 2011.

[74] Y. Xing, T. A. Vincent, M. Cole, J. W. Gardner, H. Fan, V. H. Bennetts, E. Schaffernicht, and A. J. Lilienthal, "Mobile robot multi-sensor unit for unsupervised gas discrimination in uncontrolled environments," in *2017 IEEE SENSORS*, IEEE, 2017, pp. 1–3.

[75] P. Zhu, S. Ferrari, J. Morelli, R. Linares, and B. Doerr, "Scalable gas sensing, mapping, and path planning via decentralized hilbert maps," *Sensors*, vol. 19, no. 7, p. 1524, 2019.

[76] H. Fan, V. Hernandez Bennetts, E. Schaffernicht, and A. J. Lilienthal, "Towards gas discrimination and mapping in emergency response scenarios using a mobile robot with an electronic nose," *Sensors*, vol. 19, no. 3, p. 685, 2019.

[77] P. Costa and J. Lima, *Robot@factory lite*, Last accessed 27 March 2019. [Online]. Available: `https://web.fe.up.pt/~robotica2019/index.php/pt/robot-factory-lite`.

[78] J. Lima, P. Costa, T. Brito, and L. Piardi, "Hardware-in-the-loop simulation approach for the robot at factory lite competition proposal," in *IEEE International Conference on Autonomous Robot Systems and Competitions (ICARSC)*, 2019, pp. 1–6.

[79] S. P. de Robótica, *Festival nacional de robótica*, Last accessed 12 July 2019. [Online]. Available: `http://www.sprobotica.pt/index.php?option=com_content&view=article&id=108&Itemid=62`.

[80] ——, *Festival nacional de robótica 2019, gondomar 24 a 28 de abril*, Last accessed 12 July 2019. [Online]. Available: `http://www.sprobotica.pt/index.php?option=com_content&view=article&id=120:festival-nacional-de-robotica-2019-gondomar-24-a-28-de-abril&catid=1:latest-news&Itemid=70`.

[81] I. Anvari, "Non-holonomic differential drive mobile robot control & design: Critical dynamics and coupling constraints," PhD thesis, Arizona State University, 2013.

[82] MakerBeam, *Makerbeam - think build enjoy*, Last accessed 12 July 2019. [Online]. Available: `https://www.makerbeam.com/`.

[83] R. P. Foundation, *Raspberry pi 3 model b*, Last accessed 3 June 2019. [Online]. Available: `https://www.raspberrypi.org/products/raspberry-pi-3-model-b/`.

[84] Arduino, *Arduino uno rev3*, Last accessed 30 June 2019. [Online]. Available: `https://store.arduino.cc/arduino-uno-rev3`.

[85] Pozyx, *Pozyx ultra-wideband*, Last accessed 3 June 2019. [Online]. Available: `https://www.pozyx.io/`.

[86] Arduino, *Arduino nano*, Last accessed 3 June 2019. [Online]. Available: `https://store.arduino.cc/arduino-nano`.

[87] L. M. Engineers, *How mq2 gas/smoke sensor works?* Last accessed 3 June 2019. [Online]. Available: `https://lastminuteengineers.com/mq2-gas-senser-arduino-tutorial/`.

[88] 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.

[89] Hokuyo, *Specifications*, Last accessed 22 June 2019. [Online]. Available: `https://www.hokuyo-aut.jp/search/single.php?serial=166`.

[90] G. Klancar, A. Zdesar, S. Blazic, and I. Skrjanc, *Wheeled mobile robotics: from fundamentals towards autonomous systems*. Butterworth-Heinemann, 2017.

[91] 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.

[92] J. Lima, J. Gonçalves, and P. J. Costa, "Modeling of a low cost laser scanner sensor," in *CONTROLO'2014–Proceedings of the 11th Portuguese Conference on Automatic Control*, Springer, 2015, pp. 697–705.

[93] H. J. Heijmans, *Morphological image operators*. Academic Press Boston, 1994, vol. 4.

[94] T. S. Sheikh and I. M. Afanasyev, "Stereo vision-based optimal path planning with stochastic maps for mobile robot navigation," in *International Conference on Intelligent Autonomous Systems*, Springer, 2018, pp. 40–55.

[95] J. Crank *et al.*, *The mathematics of diffusion*. Oxford university press, 1979.

[96] R. Belwariar, *A* search algorithm*, Last accessed 22 June 2019. [Online]. Available: `https://www.geeksforgeeks.org/a-search-algorithm/`.

[97] I. Noreen, A. Khan, and Z. Habib, "A comparison of rrt, rrt* and rrt*-smart path planning algorithms," *International Journal of Computer Science and Network Security (IJCSNS)*, vol. 16, no. 10, p. 20, 2016.

[98] S. Bruno and K. Oussama, *Springer handbook of robotics*, 2008.

[99] P33a, *Simtwo*, Last accessed 3 July 2019. [Online]. Available: `https://github.com/P33a/SimTwo`.

[100] J. Lima, P. Costa, P. Costa, L. Eckert, L. Piardi, A. Moreira, and A. Nakano, “A* search algorithm optimization path planning in mobile robots scenarios,” In International Conference of Numerical Analysis and Applied Mathematics, 2018.

[101] 5. R. S. Team, *5dpo components for lazarus*, Last accessed 3 July 2019. [Online]. Available: `https://sourceforge.net/projects/sdpo-cl/`.

Appendix A

First Place Certificate of Robot@Factory Lite Competition

FESTIVAL NACIONAL DE ROBOTICA 2019 GONDOMAR

CERTIFICATE OF AWARD

IPB@Factory

Instituto Politécnico de Bragança

1st Place

Robot@Factory Lite

Robótica 2019 - Portuguese Robotics Open / Festival Nacional de Robótica

held in **Gondomar, Portugal, April 24-28, 2019**

The Event Chairs:

[illegible]
António Paulo Moreira

[illegible]
Luís Paulo Reis

U.PORTO FEUP FACULDADE DE ENGENHARIA UNIVERSIDADE DO PORTO · INESCTEC TECNOLOGIA E CIÊNCIA | LABORATÓRIO ASSOCIADO · SPR · CIÊNCIA VIVA · GONDOMAR MUNICÍPIO DE GONDOMAR · AGRUPAMENTO de ESCOLAS nº 1 de GONDOMAR · Colégio Paulo VI

isep Instituto Superior de Engenharia do Porto · LIACC · CISTER · RoboCup · utad · universidade de aveiro · ipb INSTITUTO POLITÉCNICO DE BRAGANÇA · Academia de Robótica · anpri Associação Nacional de Professores de Informática

altice · controlar innovating industry · botnroll.com · JPM INDUSTRY INTRALOGISTICS MADE EASY · FFONSECA SOLUÇÕES DE VANGUARDA · ABB · Alberto Sá Lda. sá-ber · OMRON · igus · BTL

Appendix B

Publications

L. Eckert, J. Lima, L. Piardi, J. Braun, P. Costa, A. Valente, and A. Nakano, “Micromouse robot prototyping and development tools,” 2019.

Paper Submitted and Accepted

J. Braun, T. Brito, J. Lima, P. Costa, P. Costa, and A. Nakano, “A comparison of A* and RRT* algorithms with dynamic and real time constraint scenarios for mobile robots,” Springer, 2019.

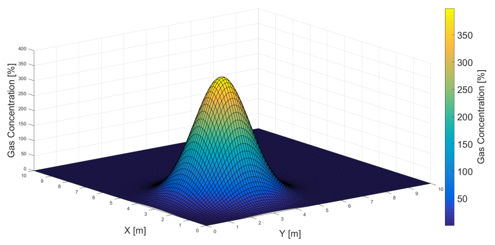
Papers Submitted and Awaiting Evaluation

J. Braun, L. Piardi, T. Brito, J. Lima, A. Pereira, P. Costa, and A. Nakano, “Indoor environment monitoring in search of gas leakage by mobile robot,” Springer, 2019.
J. Braun, L. Fernandes, T. Moya, V. Oliveira, T. Brito, J. Lima, and P. Costa, “Robot@factory lite: An educational approach for the competition with simulated and real environment,” Springer, 2019.
T. Brito, J. Lima, P. Costa, V. Matellán, and J. Braun, “Collision avoidance system with obstacles and humans to collaborative robots arms based on RGB-D data,” Springer, 2019.

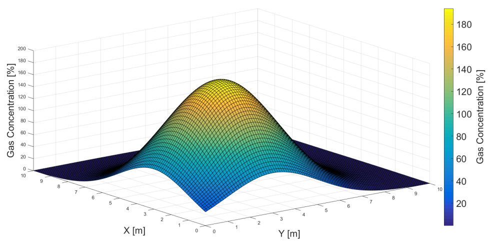
Appendix C

Gas Model Distribution with Time Dependency

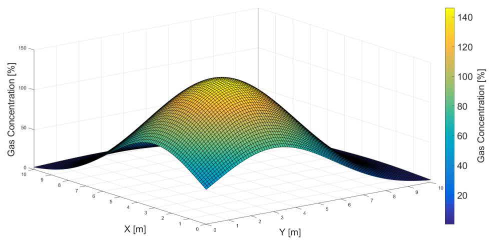
The value $D \times t$ goes from 0.5 to 15 linearly distributed in eleven figures . The gas distribution is centered at $[x, y] = [3, 3]$.



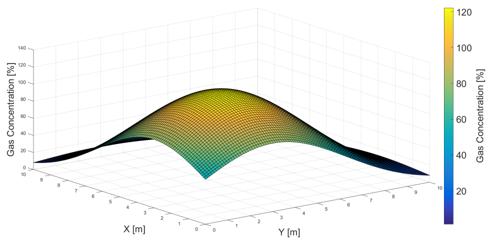
(a) Figure 1.



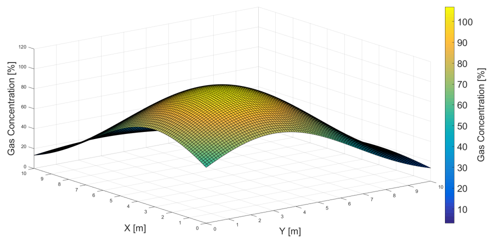
(b) Figure 2.



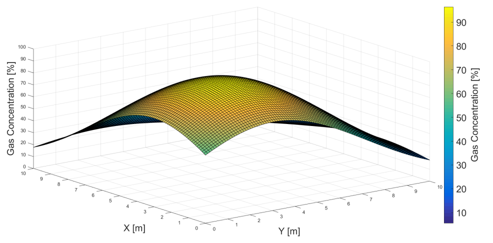
(c) Figure 3.



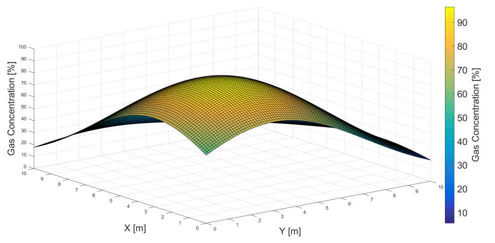
(d) Figure 4.



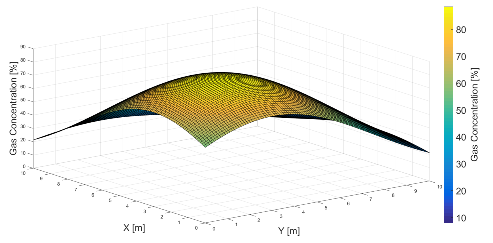
(e) Figure 5.



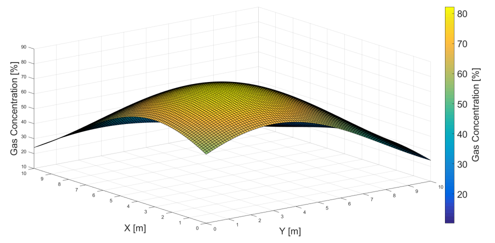
(f) Figure 6.



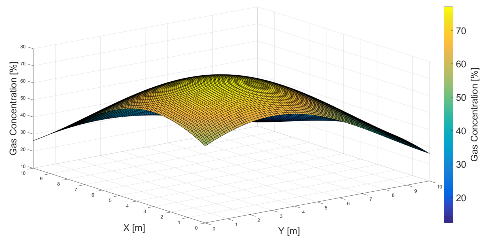
(g) Figure 7.



(h) Figure 8.



(i) Figure 9.



(j) Figure 10.

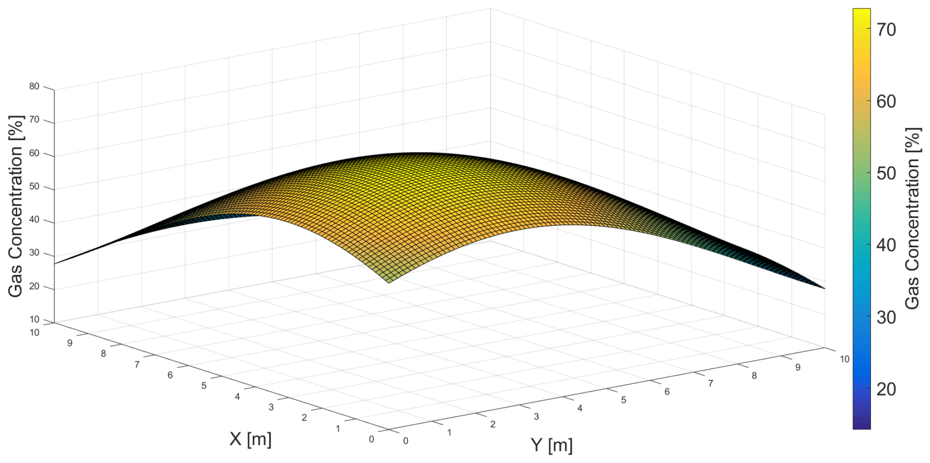


Figure C.2: Figure 11.

Appendix D

Assembly Manual of Robot@Factory Lite

Robot@Factory

Lite

Provided kit assembly manual

PORTO

2019

JOÃO AFONSO BRAUN NETO

Provided kit assembly manual

PORTO

2019

Manual made to help the robot’s assembly to the participants of the Robot@Factory Lite competition, that have chosen to use the designed kit by the competition organizers. The kit with the designed parts, bill of materials, electric schematic, connections and simulator with pre-programed hardware in the loop can be found at:
https://github.com/P33a/RobotAtFactoryLite.

Table of Contents

List of Figures

List of Materials

Components	Quantities
DC Motor	2
Wheels	2
RFID Reader	1
Line Sensor	1
Electromagnet	1
Caster Wheel	1
Battery support	2
Motors driver	1
Arduino Nano	1
Arduino Nano shield	1
Lithium battery	2
Step down converter	1
Micro switch	1
Connection Wires	x
Battery Charger	1
Wood screw 2.5X12 mm	28
Plain female screw 3X10 mm	4
Wood screw 3X15 mm	4
Female screw 3X30 mm	4
Female screw 3X20 mm	2
Female screw 3X6 mm	1
Wire wrap	2
Screw nut 3 mm	11
Power Button/Switch	1
Heat Shrink Sleeves	x
FDP3682	1
3D Printed Parts	Quantities
raflite_base	1
raflite_RFID	1
raflite_RFID_support	1
raflite_spacer	2
yellow_motor_support	2
motor_driver_support	1/opt

ROBOT’S ASSEMBLY

This manual will cover the assembly of the robot step by step. Note that, maybe it will be necessary to widen up just a little bit the holes in the parts because of imprecision from the 3D printer. After the material needed is gathered and printed, the first thing to do is to screw four pillars to the base of the robot. It will be needed four 2.5X12 mm wood screws. After that, screw the front sensors support to the front of the base. These steps can be seen in Figures 1, 2, 3 and 4.

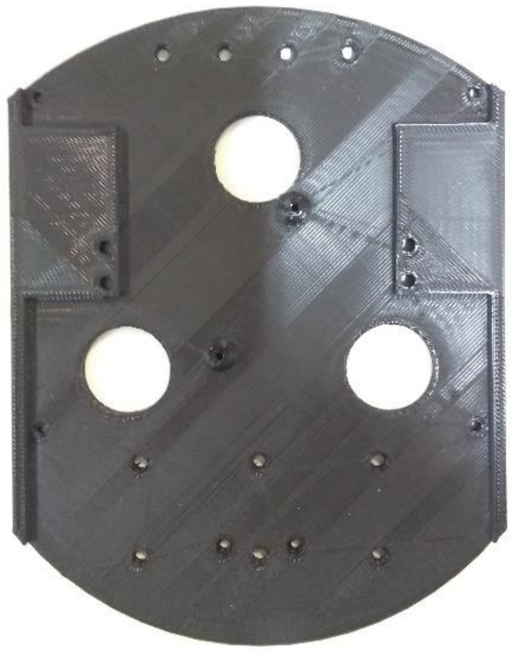


Figure 1. Robot’s base.

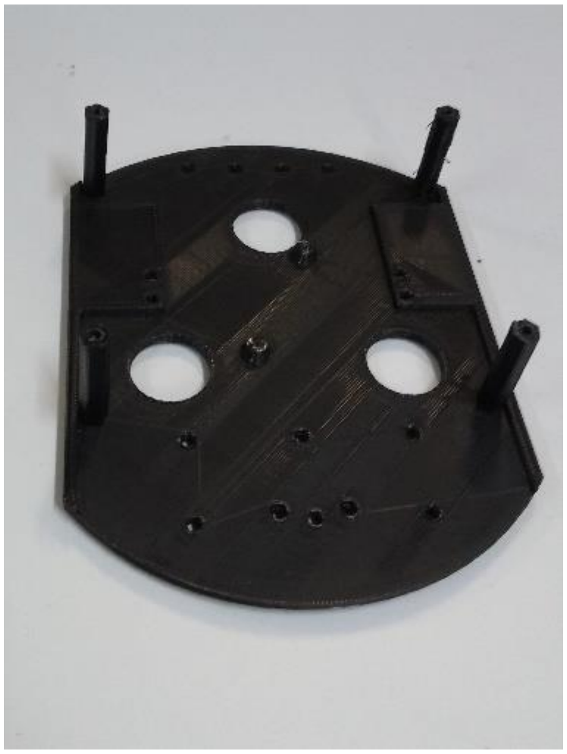


Figure 2. Four pillars positions.

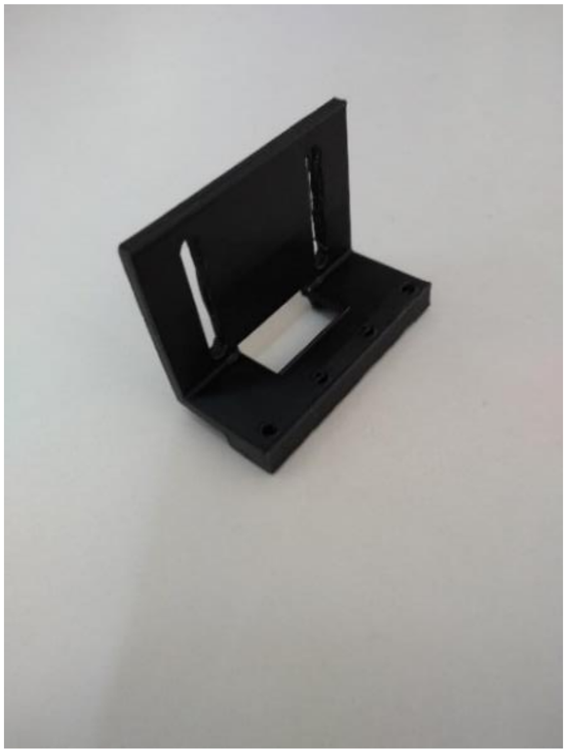


Figure 3. RFID support assembly.

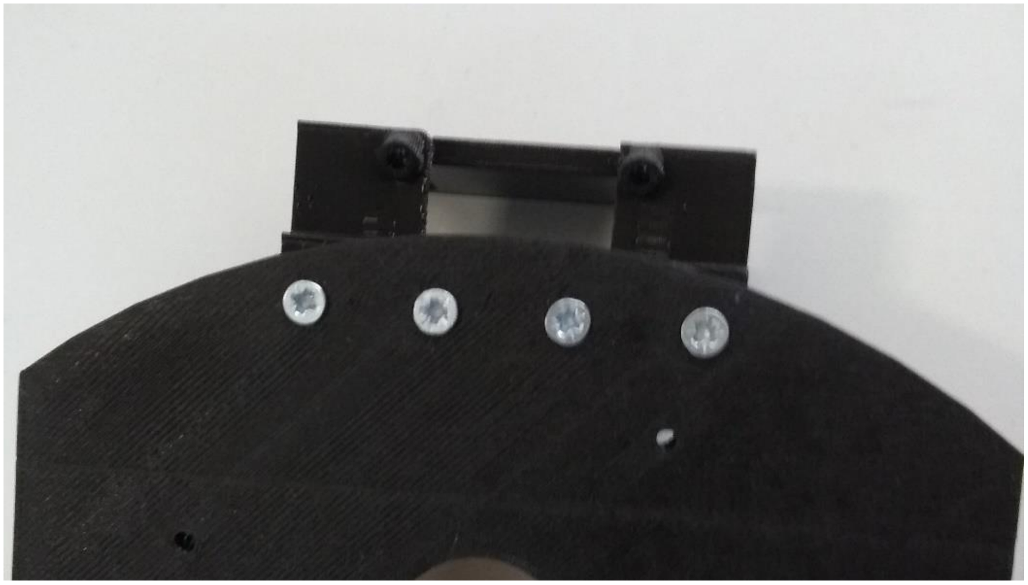


Figure 4. Robot’s base bottom view with the front support.

The next step is to screw the microcontroller Arduino Uno to the upper robot’s base. For this procedure it is necessary four 2.5X12 mm wood screws. See Figures 5 and 6.

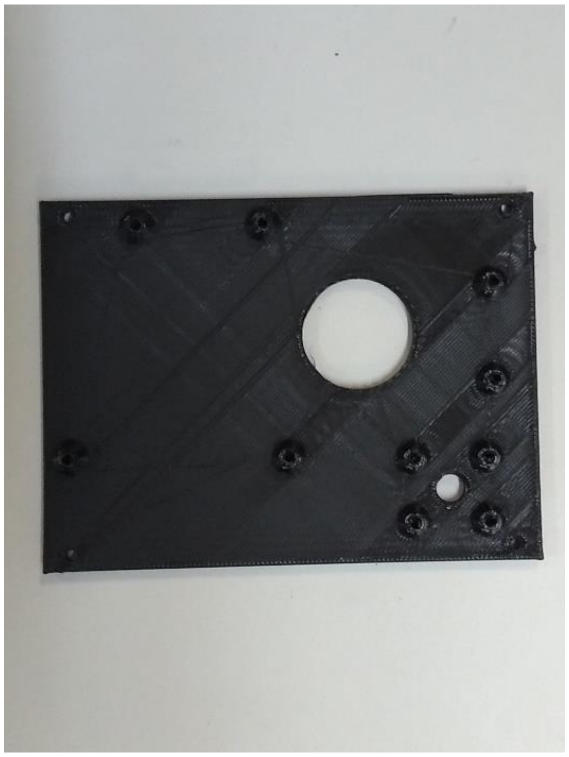


Figure 5. Robot's upper base.

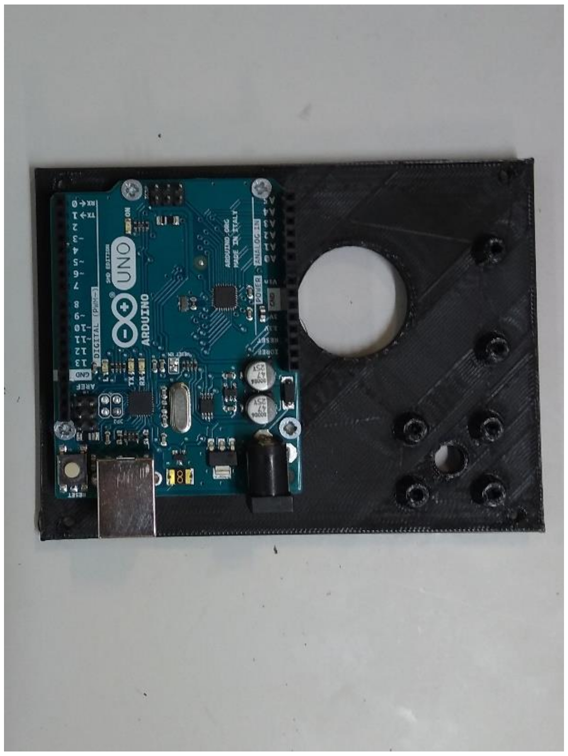


Figure 6. Arduino’s assembly to the base.

Now, the task is to screw the RFID integrated circuit to the RFID part with four 2.5X12 mm wood screws. It's recommended to weld male pin headers to the IC. Soon after, with the spacers in the list of materials section, screw the RFID part with two female 3X20 mm screws coupled with the correspondent screw-nuts to the RFID support part seen in Figure 3. This task can be seen in Figures 7 and 8.

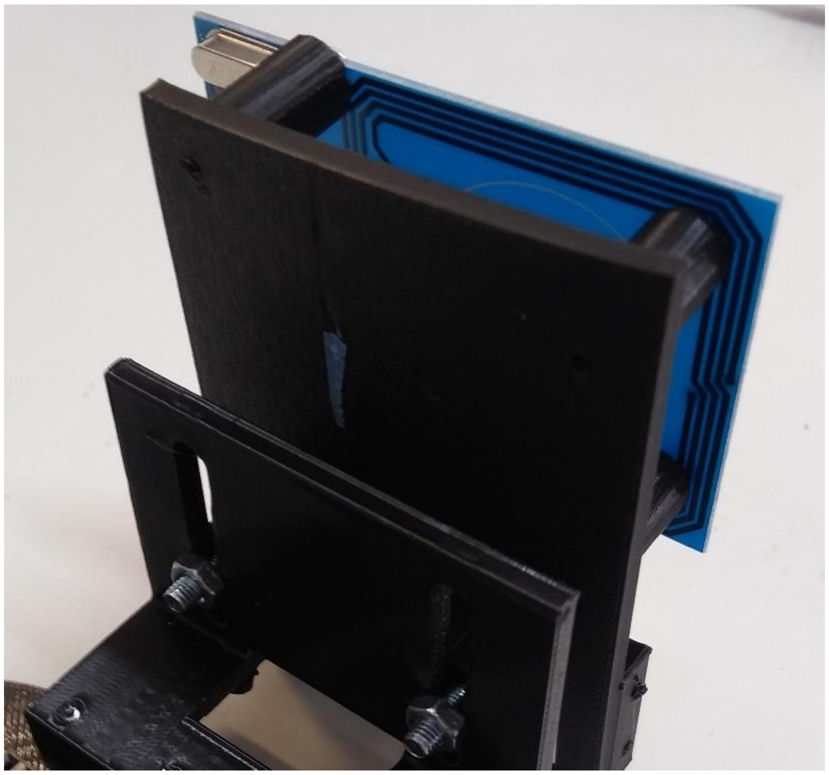


Figure 7. The assembly of RFID IC and support.

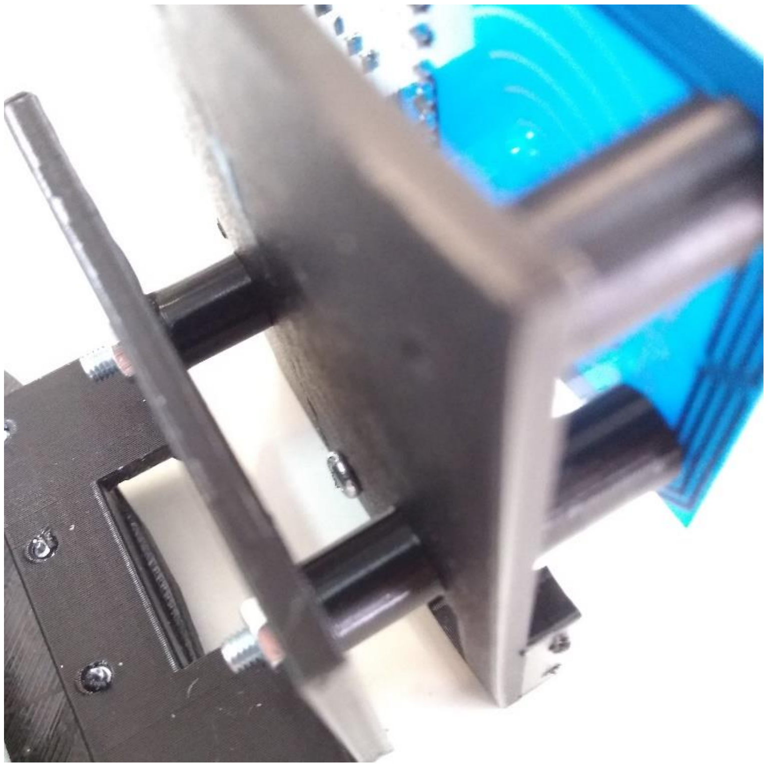


Figure 8. Spacers and female screws with screw-nuts.

After that, screw the microswitch with roller to the RFID part with two 2.5x12 mm wood screws. It can be seen in Figures 9 and 10. It's necessary to weld the two wires with heat shrink sleeves, this way there will be no danger to short circuits. Additionally, as shown in the schematic, the yellow wire has a 1 kΩ resistor in series, as can be seen in Figure 11 pointed by the green arrow. In addition, pay attention that there's 2 output wires welded together after the resistor. This was made because we are using the same digital pin both for the microswitch and the MOSI RFID pin, in this way, later in the connection phases the black wire will go to the eleventh digital pin in the Arduino and the blue wire will go to the RFID MOSI pin. The reason for the resistor is that, when the switch is activated, without the resistor, we would not be able to communicate with the RFID Reader. However, with the resistor, the pin 11 and MOSI will not be grounded.

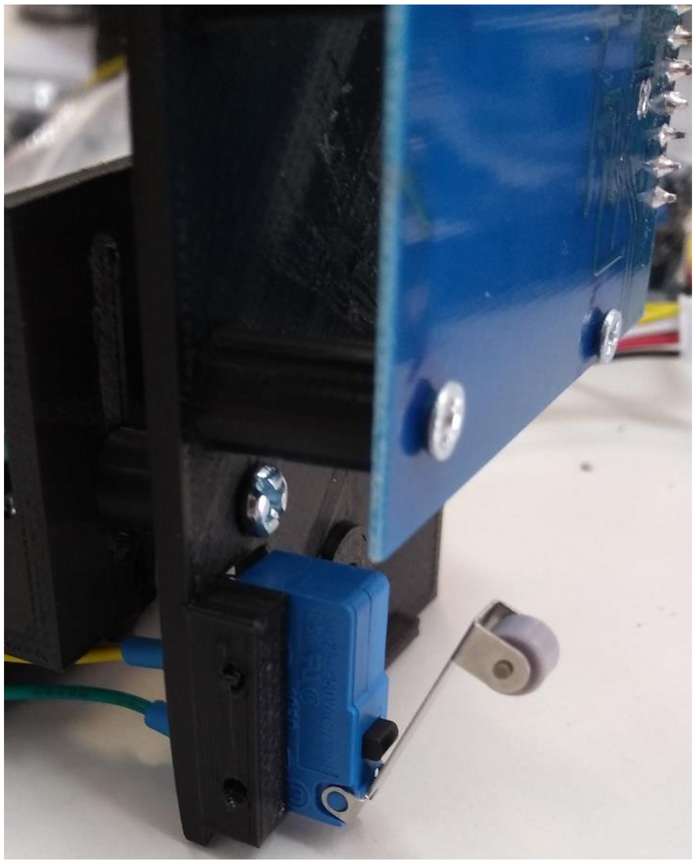


Figure 9. Microswitch position.

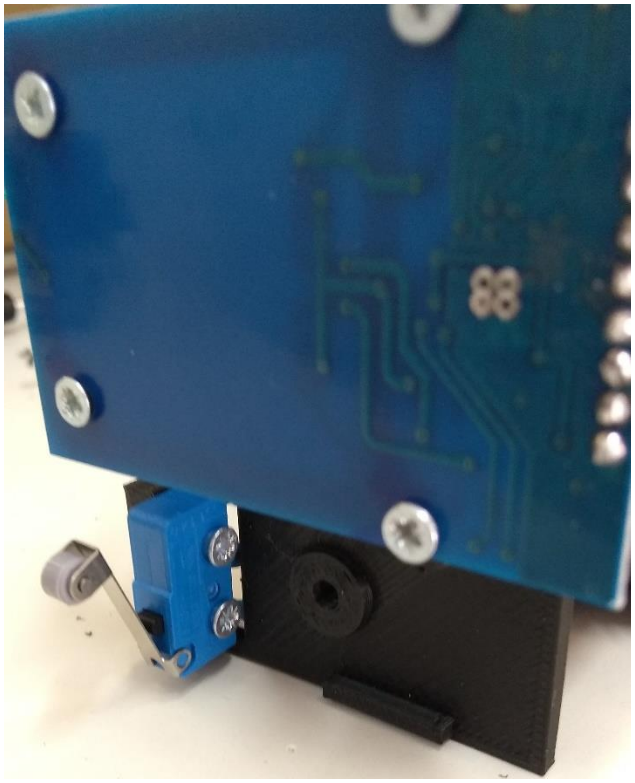


Figure 10. Microswitch assembly.

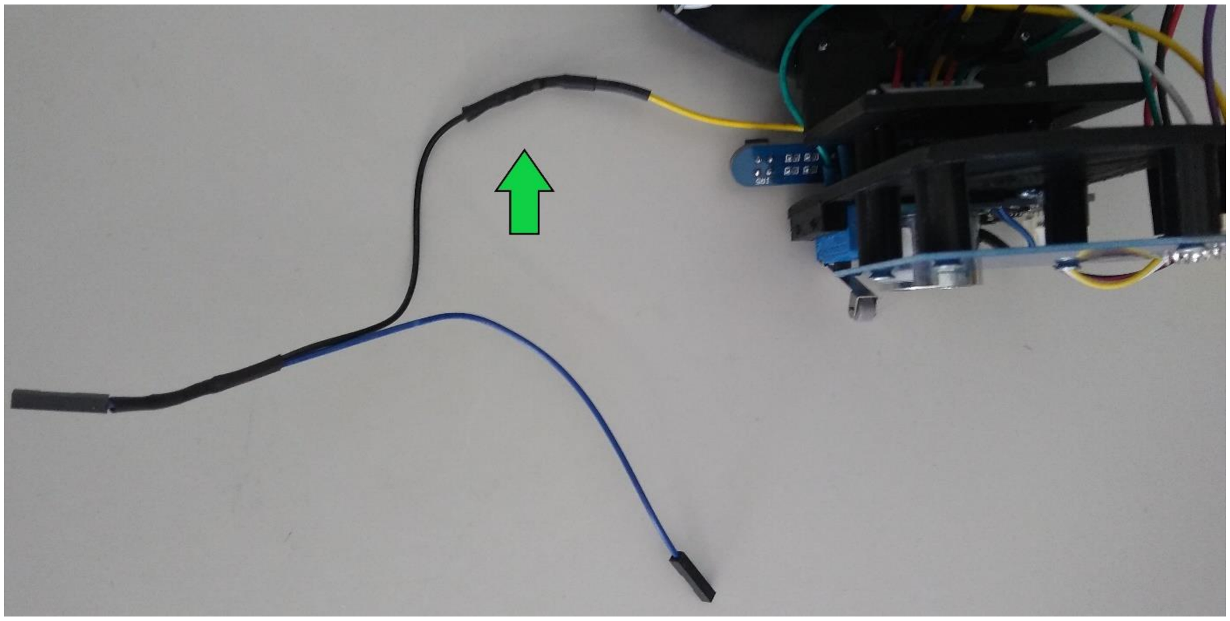


Figure 11. Welded wires of the switch.

Next, to assemble the electromagnet, first remove the screw from the IC, as seen in Figure 12.

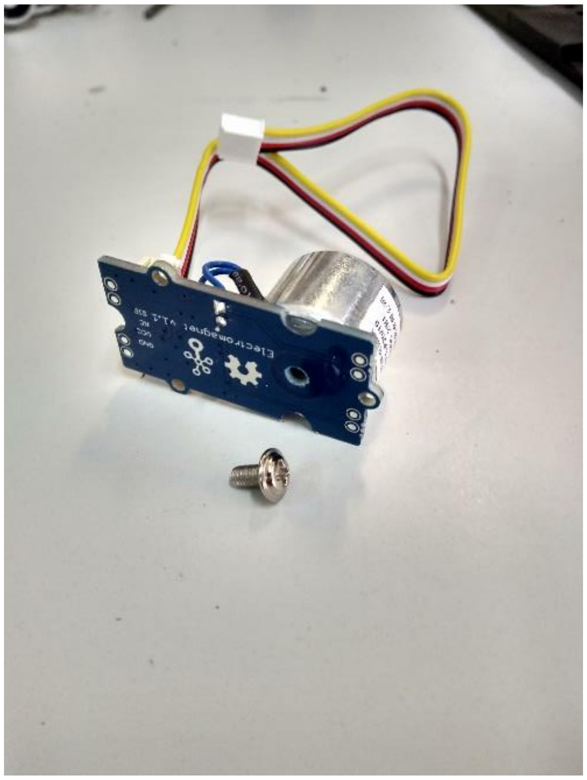


Figure 12. Electromagnet.

After this, screw the IC to the RFID part as seen in Figure 13.

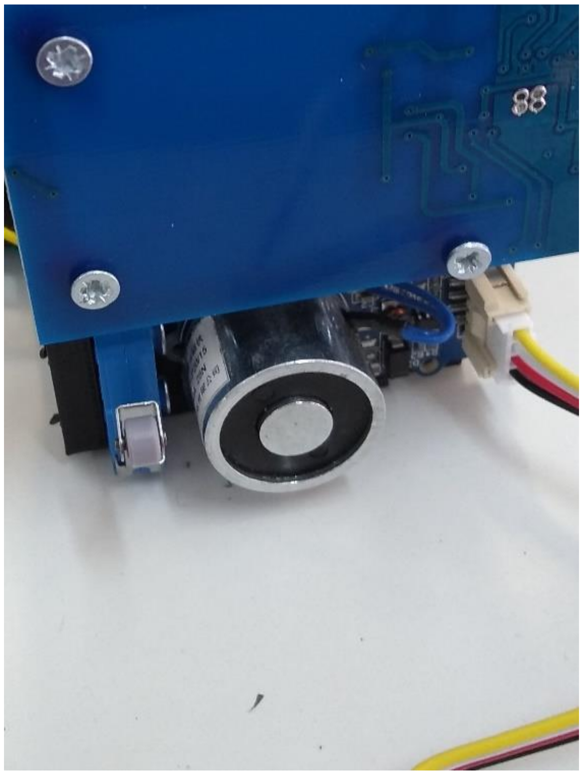


Figure 13. Electromagnet assembly.

Afterwards, screw the infrared line tracking sensor to the bottom of the RFID support part as seen in Figures 14 and 15. It is necessary two 2.5X12 mm wood screws for the task.

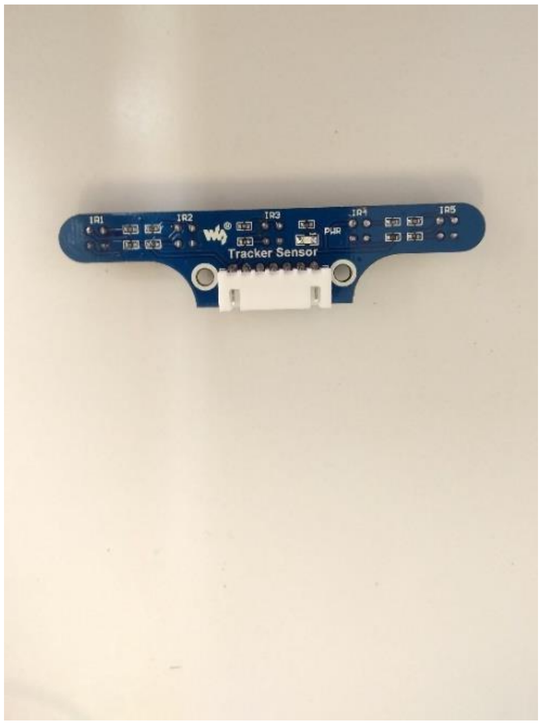


Figure 14. Infrared line tracking sensor.

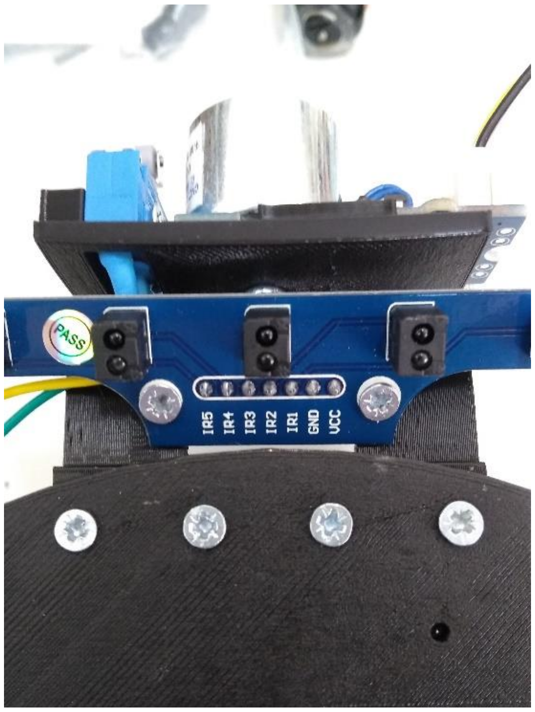


Figure 15. Bottom view of the front support.

Later, screw the motor support parts with two 3X15 mm wood screws per support. The location can be seen in Figures 16 and 17. It was used wood screws, for better attrition.

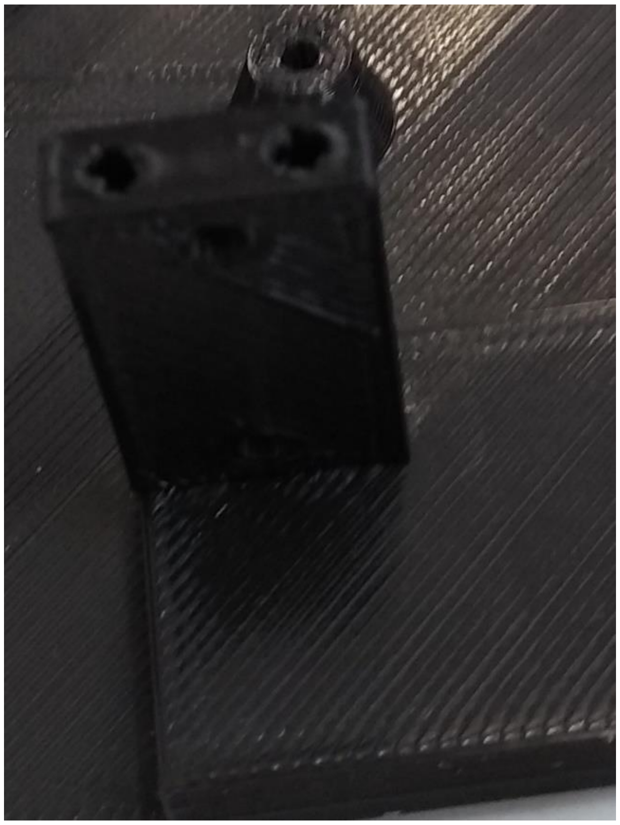


Figure 16. Motor support.

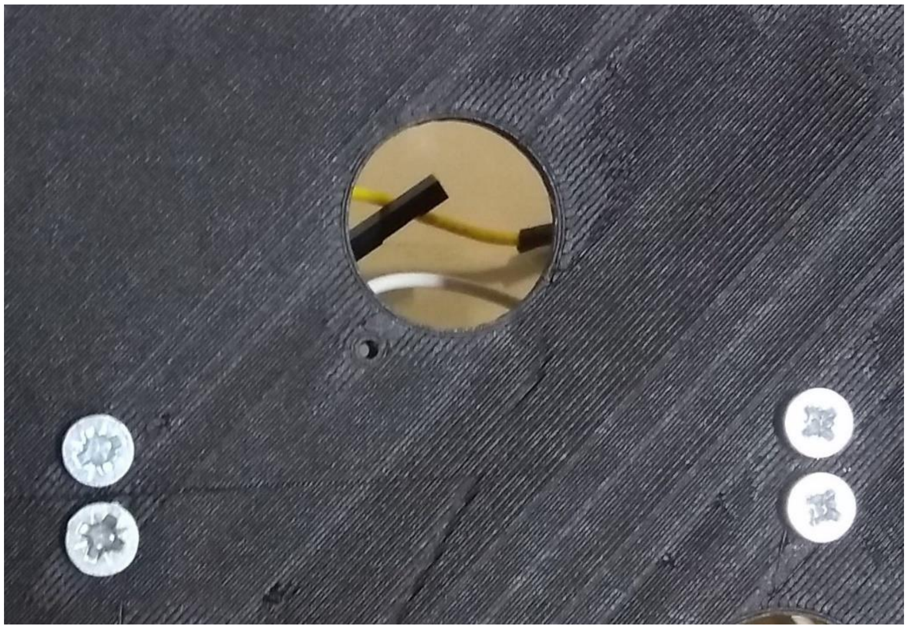


Figure 17. View of the screw position for the motor support.

The next step is to screw the battery supports to the base. It is necessary four plain 3X10 mm female screws, attach the respective screw nuts from below the base. The procedure can be seen in Figure 18. Additionally, pay attention that the batteries support is welded in series, as shown in the electric schematic.

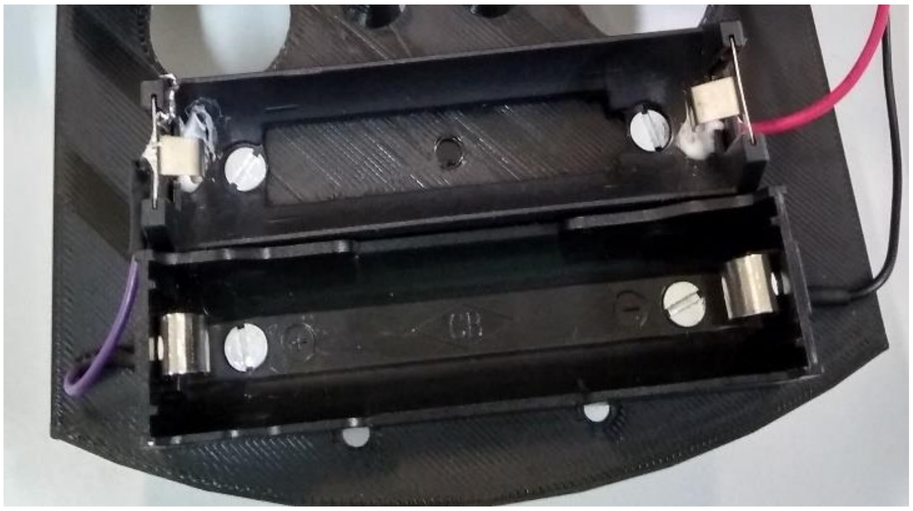


Figure 18. Batteries support.

Soon after, the figure below displays two electric schemes for the two different MOSFET channels. If you have a P channel, please follow the left scheme. In the other hand, if it is in hand a N channel field effect transistor, follow the right one. This circuit is to provide reverse electrical polarization protection to the robot. In this way, if the batteries are placed with the poles reversed, the FET will not enter the saturation state and, consequently, protecting the robot's circuits.

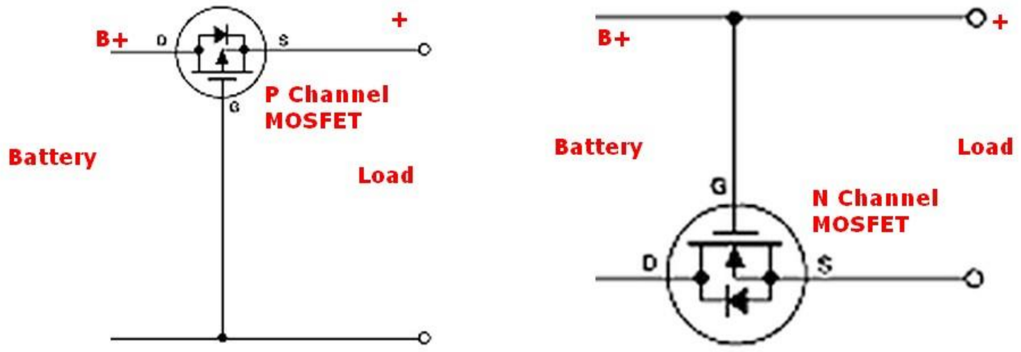


Figure 19. Electric schemes for the reverse polarization protection.

To install, make a 3 mm hole besides the battery's output and screw the transistor to the base of the robot with a 3X6 mm female screw and its respective screw nut as can be seen in Figure 20. After that, just weld the output batteries' wires as Figure 19 shows.

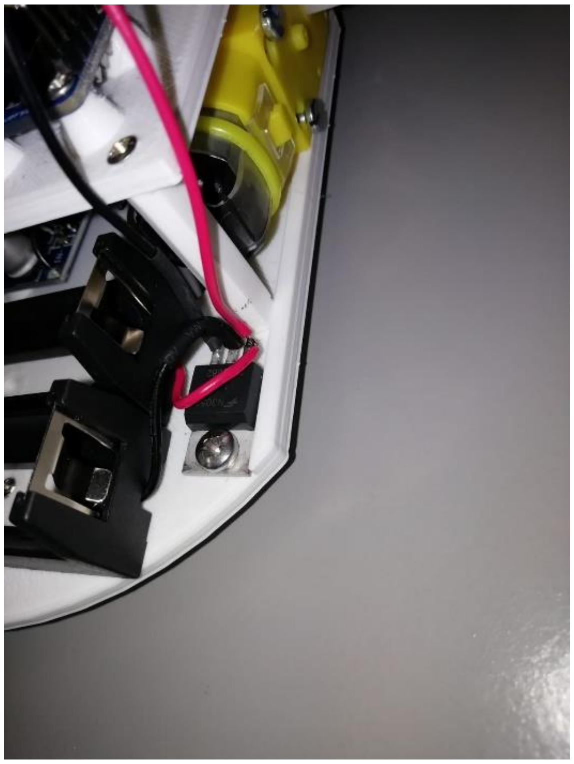


Figure 20. MOSFET's assembly.

Now, install the castor wheel in the way Figure 21 shows. The castor wheel comes with one bigger sphere and four smaller ones, two screws and the case. Just insert the bigger sphere into the case with the four smaller ones besides it. After that, just screw the case shut.

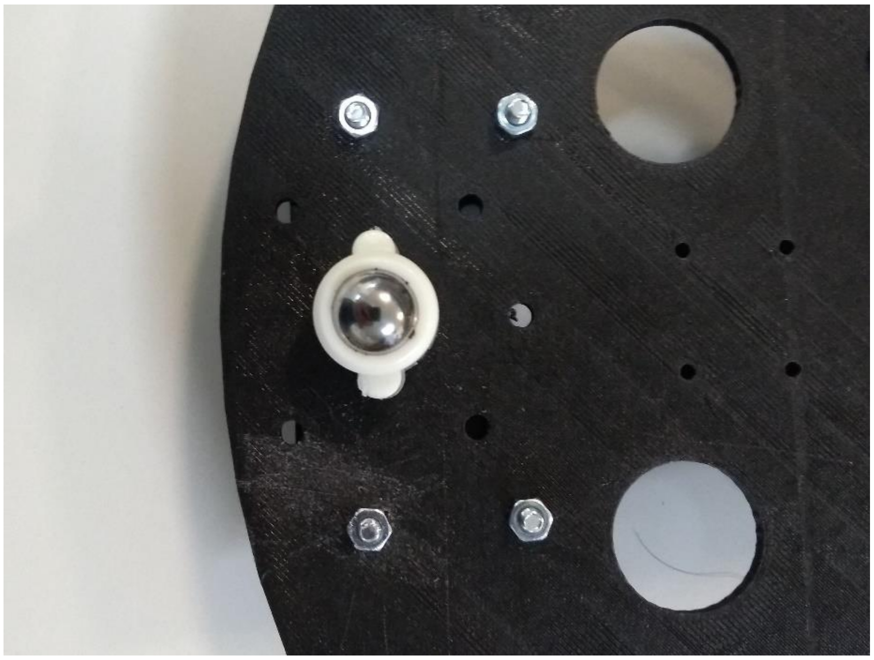


Figure 21. Bottom view of the base.

To install the motors, just use two 3X30 mm female screws with their respective screw nuts. Screw the motor to its support as shown in Figure 22. After that, just insert the wheel to the axis. Pay attention that you will need some pressure on the center of the wheel to push the fitting part to the axis. Do the same to the other side, shown in Figure 23. Use the wire wraps to tight up the motors' wires seen in both Figures below.

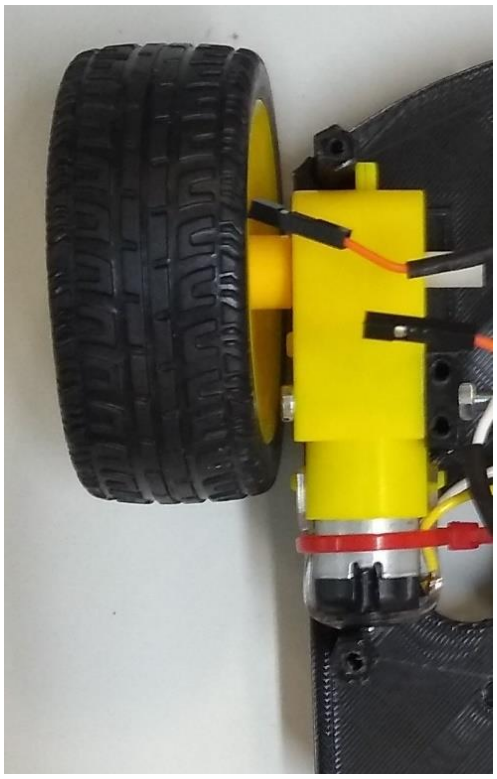


Figure 22. Left motor with wheel setting.

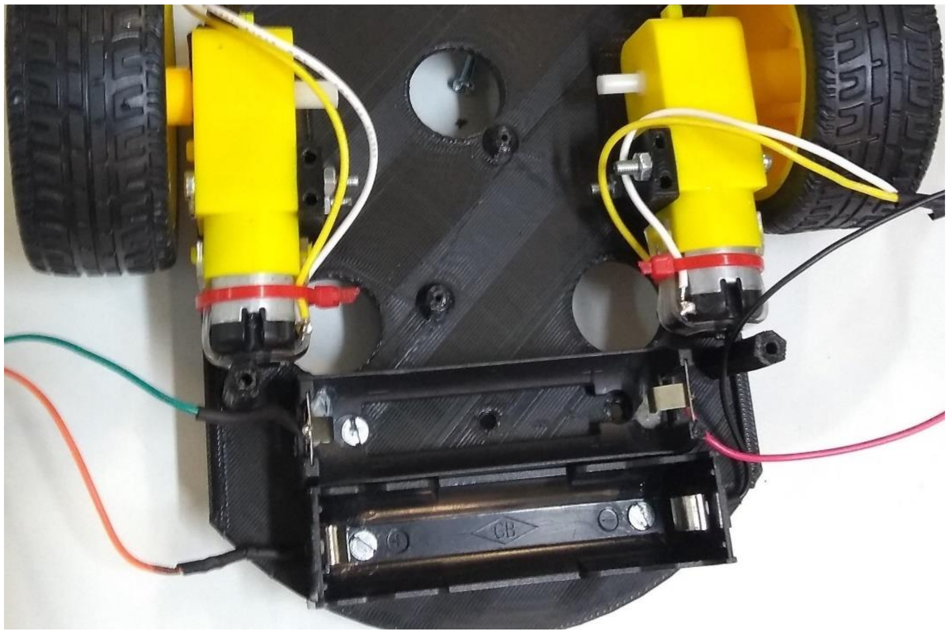


Figure 23. Top view of the robot.

To embed the dual motor driver, just screw it with two 2.5X12 mm wood screws. As we didn’t have the same model as in BoM, it was designed a support for our driver as seen in Figure 24. In addition, insert the expansions shield to the Arduino Uno as seen in Figure 25.

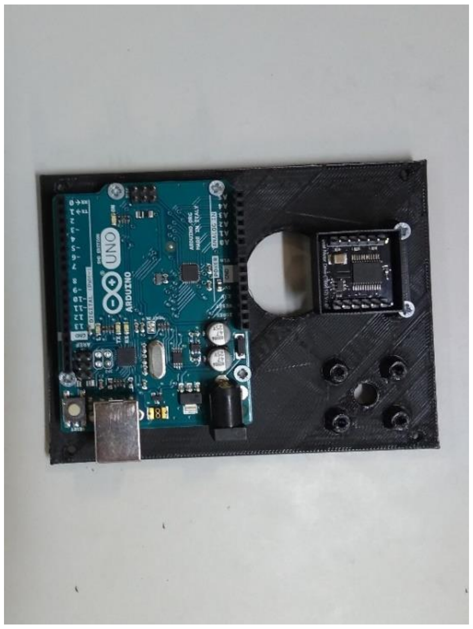


Figure 24. Dual motor driver.

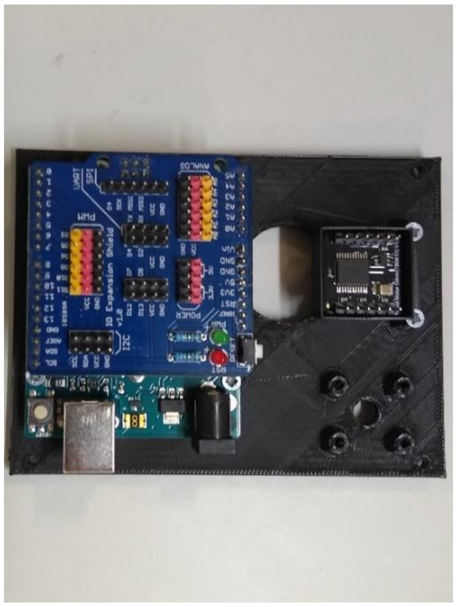


Figure 25. Expansion Shield.

Before assembling the stepdown, configure it to convert the DC input voltage to 5.2 V output. After this, the stepdown support assembly and position can be seen in Figure 26. Screw the part with two 2.5X12 mm wood screws as shown. It is recommended to first weld the input and output wires in the converter.

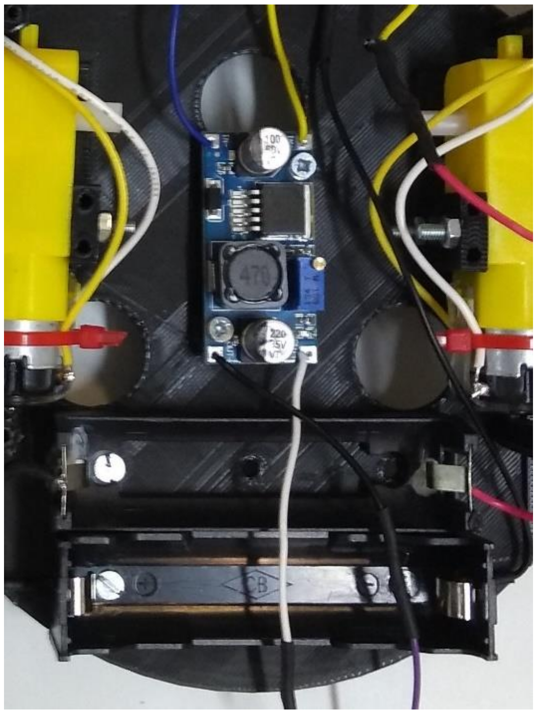


Figure 26. Stepdown converter.

Note that in Figure 27, there is more than one connector to the wire. This is done to supply the dual motor driver as well.

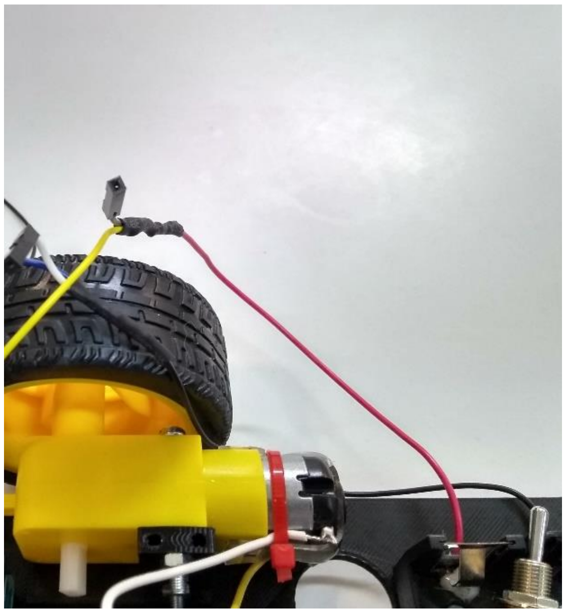


Figure 27. Battery to stepdown/driver wires.

Connect the IR sensor connector as shown in Figure 28.

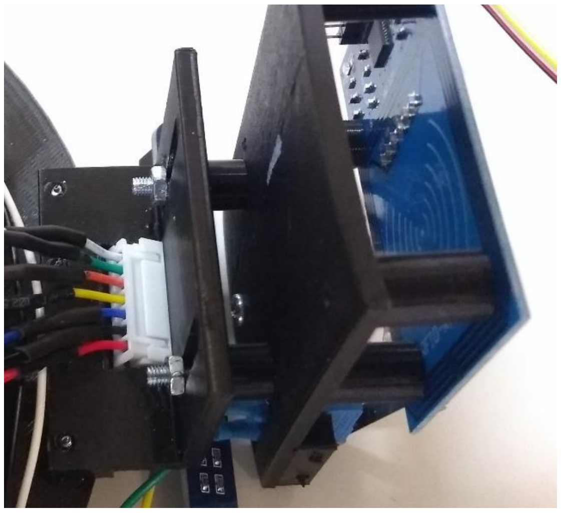


Figure 28. Infrared sensor connector.

This step is totally optional. In BoM document, there is a power button or a power switch to be bought. If it is chosen the first one, just screw with four 2.5X12 mm wood screws in the position Figure 30 shows. However, if the latter is chosen, follow this guide. The power switch and its support seen in Figures 29 and 30 are available in the kit. If you choose to install, just weld the switch in series with the batteries as can be seen in Figure 29 and install with its support as in Figure 30. In Figure 29, the spot is a bit below of the driver.

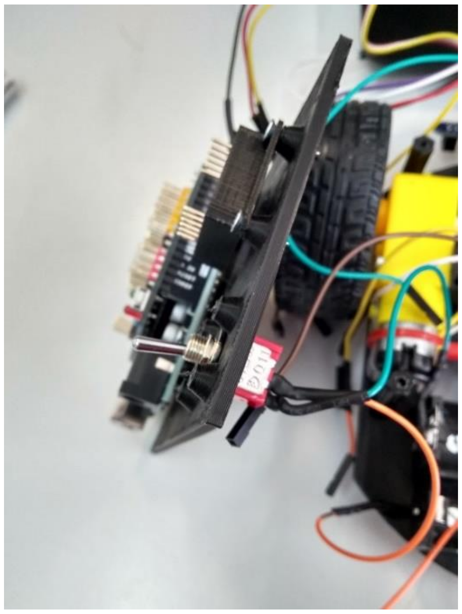


Figure 29. Power Switch.

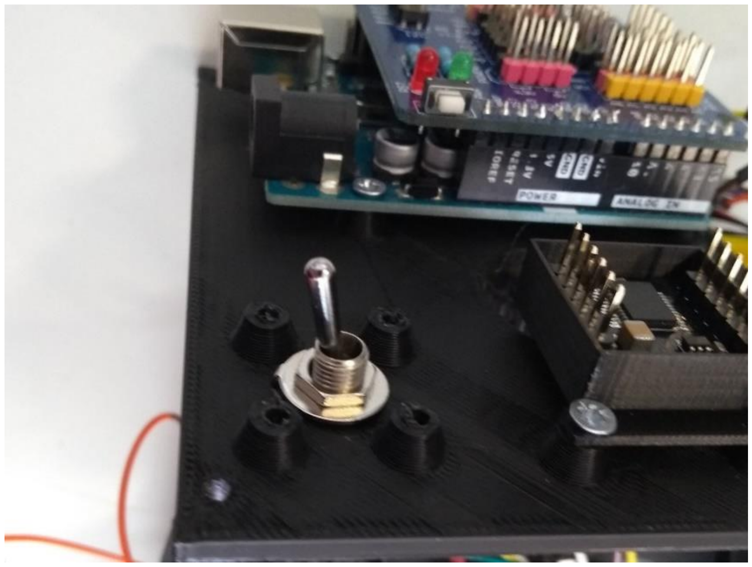


Figure 30. Power switch and its support.

After installing the switch, screw the top layer to the base of the robot using four 2.5X12 mm wood screws. This step can be seen in Figure 31. Now, the final steps are to make the correct connections to the expansion shield. Please, follow the electric schematic and the connections text file to guide you through out this phase. Remember that the RFID receptor is driven by 3.3 V not 5 V. In the other hand, both the electromagnet and the IR sensor are driven by 5V.

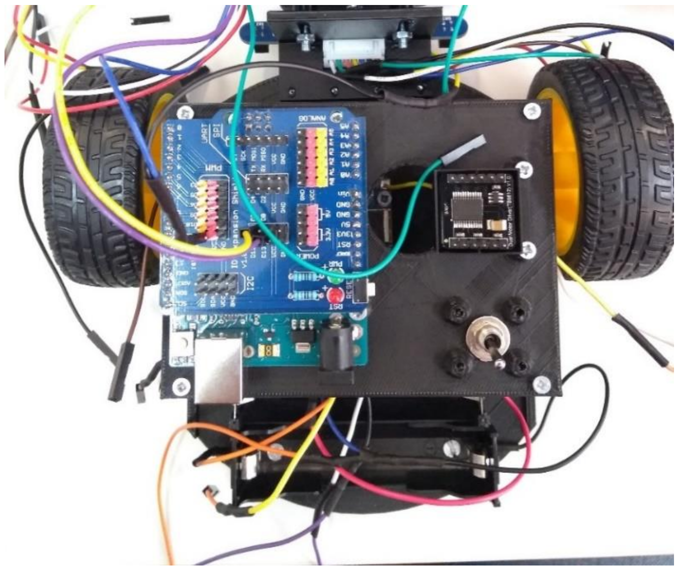


Figure 31. Top View of the Robot.

The robot after its connections, is shown in Figures 32, 33 and 34. Be cautious when connecting the VM, VCC, GND and the stepdown pins so that any component will be harmed by voltages above the secure threshold. Make sure that the output voltage of the stepdown is configured to 5.2 V. This way, the stepdown will drive the peripherals, not the Arduino.

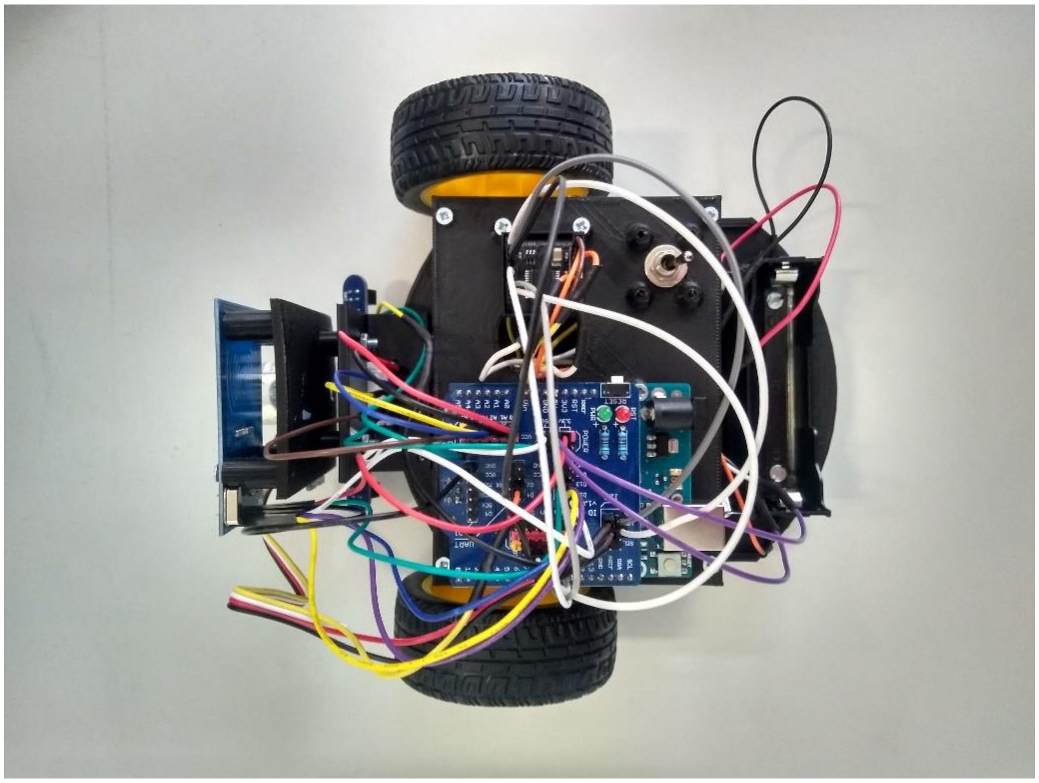


Figure 32. Robot's top view after its assembly.

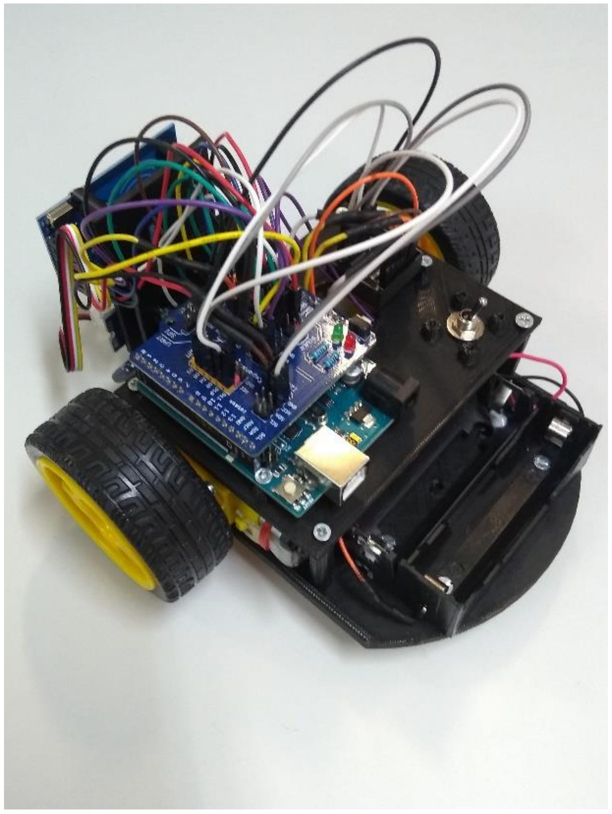


Figure 33. Isometric View of the Robot.

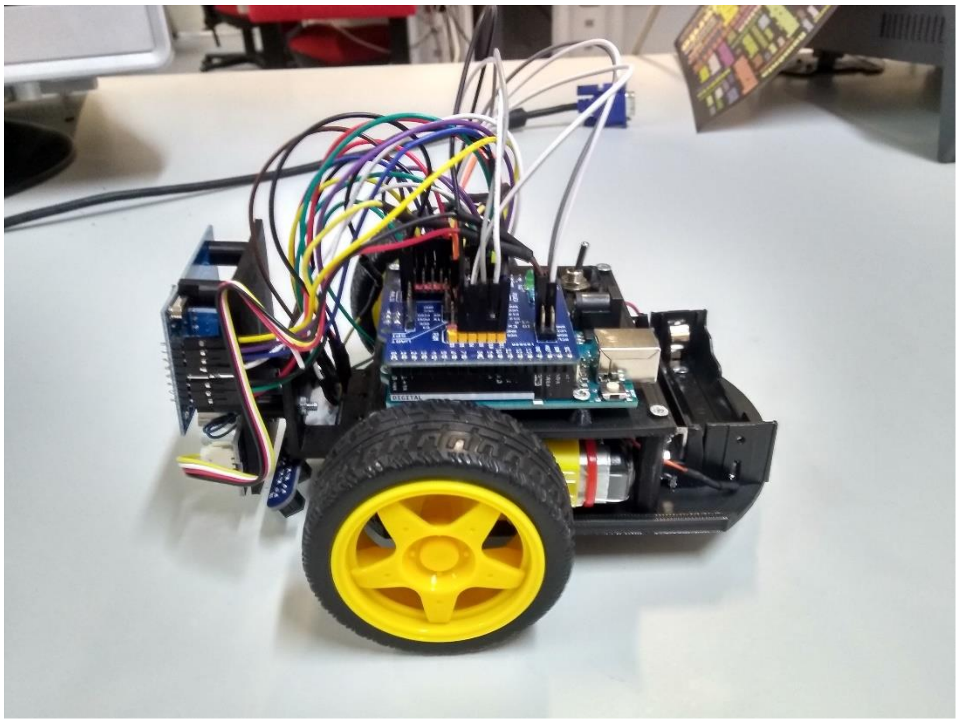


Figure 34. Robot's sideview after its assembly.

ELETRIC SCHEMATIC

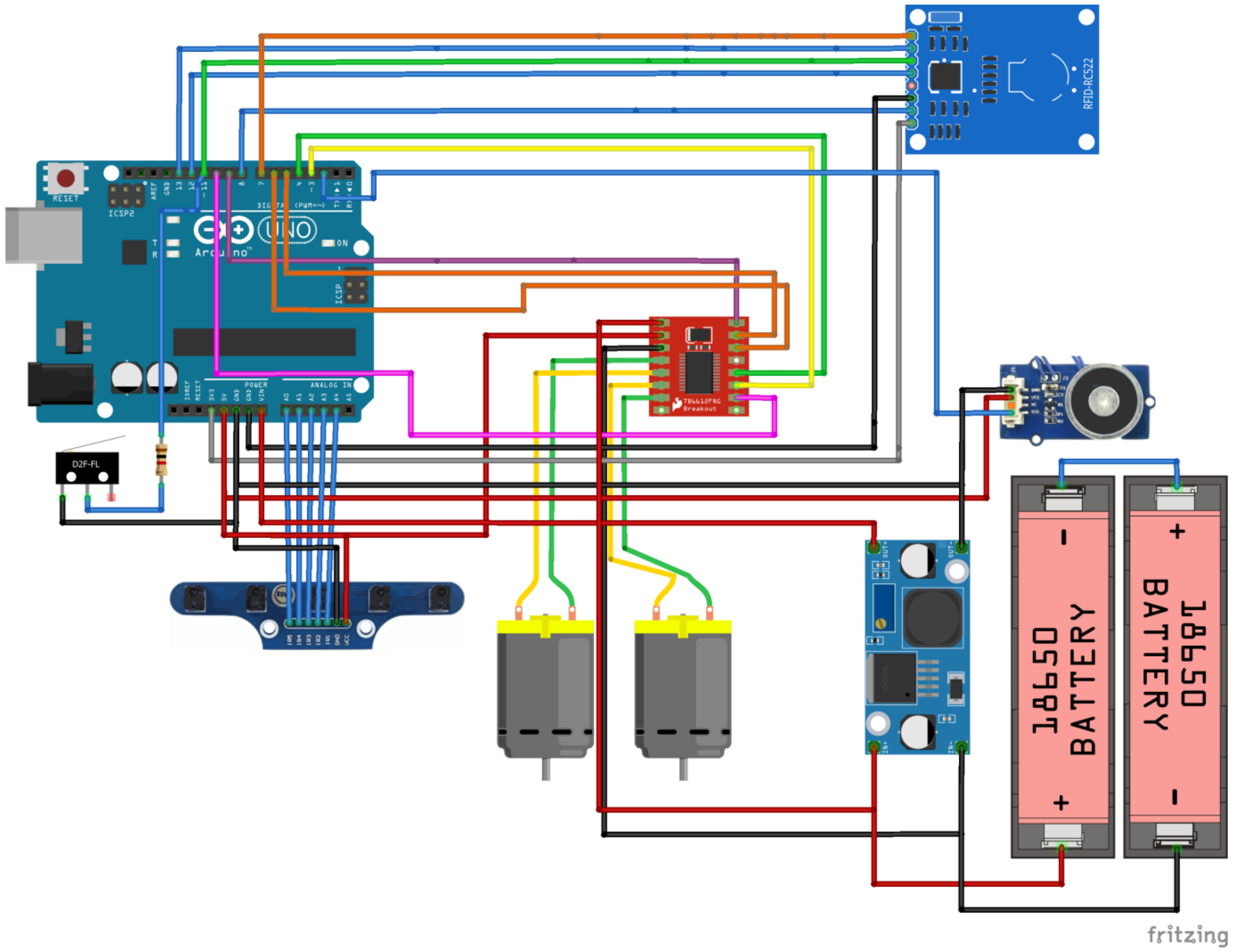


Figure 35. Electric Schematic.

Connections table

Microcontroller and its peripherals connections	
RFID	
SCK	13
MISO	12
MOSI	11
RST	8
SDA	7
IRQ	NC
GND	GND
3.3 V	3.3 V
IR Sensor	
IR5	A0
IR4	A1
IR3	A2
IR2	A3
IR1	A4
GND	GND
VCC	5 V/VCC
Motors A-Left B-Right	
PWMA	9
AIN1	6
AIN2	5
PWMB	10
BIN1	4
BIN2	3
MOTORA	Left motor wires
MOTORB	Right motor wires
VCC	5 V/VCC
VM	<12 V / Batteries wires
GND	GND
Solenoid	
SIG	2
VCC	5 V/VCC
GND	GND
Microswitch with roller	
Mid pole with resistor	11
Extremity pole	GND
Stepdown	
IN+	Battery +
IN-	Battery -
OUT+	5 V/VCC
OUT-	GND

THE SHOP FLOOR GARAGES ASSEMBLY

The garages can be assembled by mounting them together with the assists of printed connectors as seen in Figures 36 and 37. Note that, to external walls have the same width as the internal ones, adjacent walls were coupled as well in Figure 37.

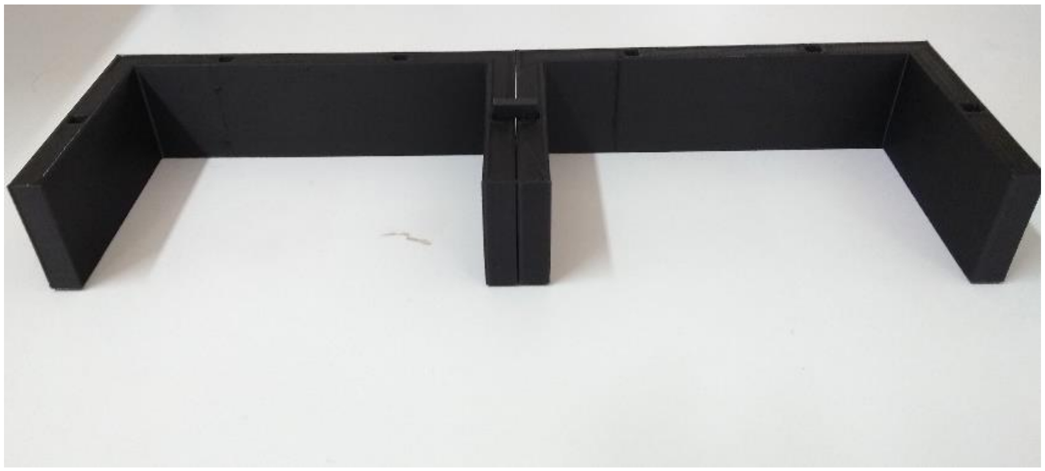


Figure 36. Garage assembly.

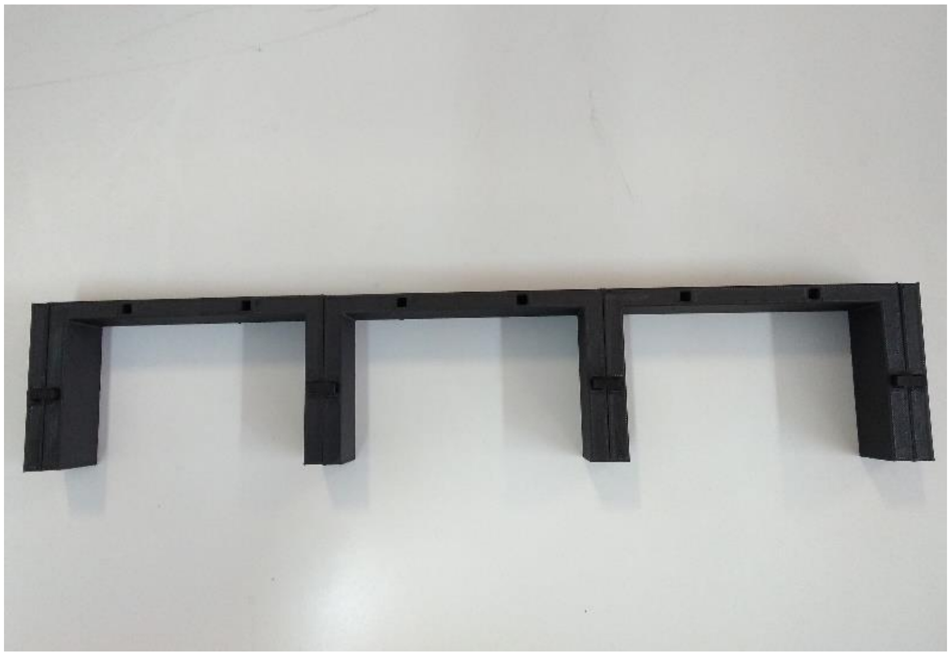


Figure 37. Three garages assembly topview.

See the garage position where the boxes will be put in Figure 38.

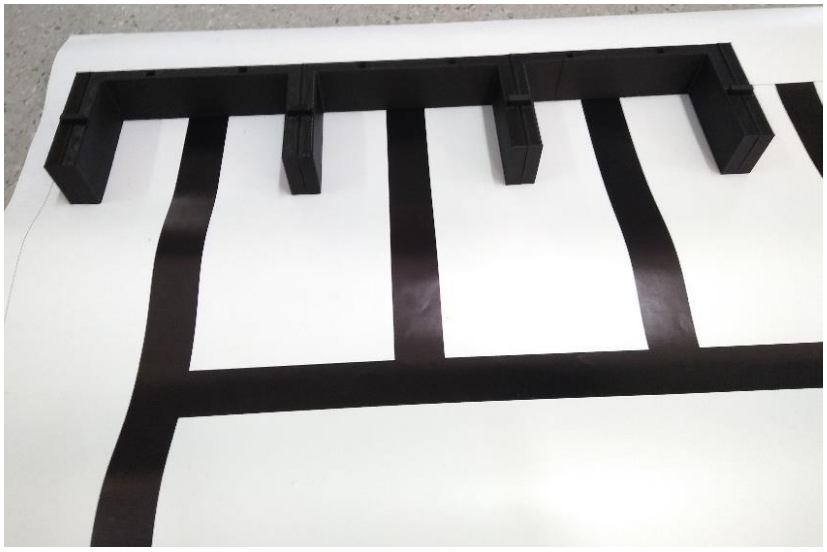


Figure 38. Garages position on the Shop Floor.

As for the machines, the assemble the parts as shown in Figure 39. The position of the machines is specified in the rules of the competition found in the site of the competition as well as in the kit from git hub. However, as can be seen in Figure 40, the position of one IN/OUT garage from machine A is displayed.

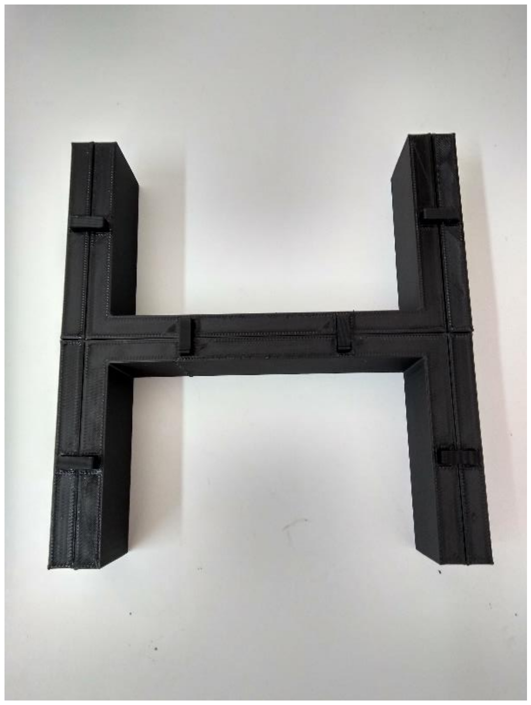


Figure 39. Part of Machine A.

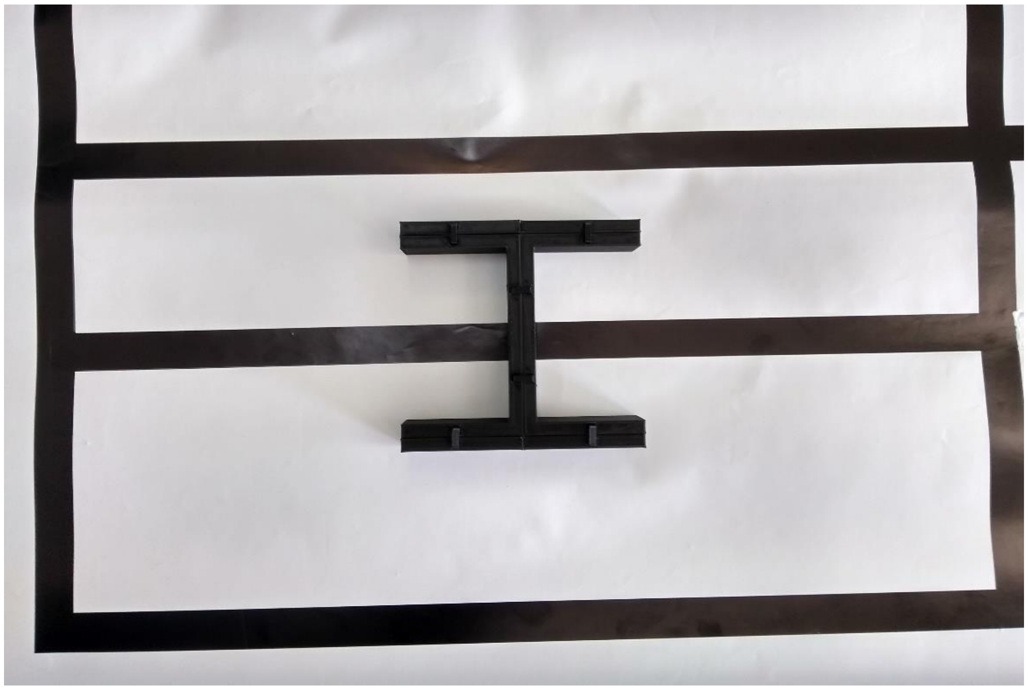


Figure 40. Position of Machine A.