Robot@Factory Lite: An enhancement for the mobile robot using quadrature encoders

Vitor Felipe Alves de Oliveira

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

Work supervised by:

PhD. José Luis Sousa Magalhães Lima

PhD. José Alexandre Carvalho Gonçalves

PhD. César Rafael Claure Torrico

Bragança

2019-2020

Robot@Factory Lite: An enhancement for the mobile robot using quadrature encoders

Vitor Felipe Alves de Oliveira

Dissertation presented to the School of Technology and Management of Bragança to obtain the Master Degree in Engenharia Industrial according to the double diploma agreement with UTFPR.

Work supervised by:

PhD. José Luis Sousa Magalhães Lima

PhD. José Alexandre Carvalho Gonçalves

PhD. César Rafael Claure Torrico

Bragança

2019-2020

Acknowledgement

I would like to express my thanks to PhD José Luis Lima, supervisor of this work, for the dedication, guidance, incentive and opportunity to develop this work, and articles.

To PhD José Alexandre Gonçalves, co-supervisor of this work from IPB.

To PhD César Rafael Torrico for accepting to be the distance co-supervisor from UTFPR in Brazil.

To my friends and laboratory colleagues, for sharing their knowledge and good conversations in the coffee breaks.

I am thankful to IPB and UTFPR for providing this opportunity in this double diploma program.

Finally, a special thanks to my parents and family for the emotional and financial support in this experience, without them this work would not be possible.

Resumo

As competições robóticas estão crescendo a cada ano, trazendo benefícios como soluções para problemas do dia-a-dia, ajudando na educação e aumentando o interesse de crianças e adolescentes em áreas de viés tecnológico. Este trabalho foca-se na competição Robot@Factory Lite apresentando melhorias para o robô utilizado na competição. Após participar da primeira edição da competição, foram constatados alguns defeitos no funcionamento do robô que atrapalhavam o desempenho das equipes, visto que todas as equipes optaram por utilizar o mesmo modelo de robô. Foi desenvolvido um robô utilizando encoders para substituir a movimentação baseada em tempo por uma baseada em distância. O controle de velocidade é feito através de um controlador PID. Um sistema de odometria foi implementado para monitorar a posição do robô. Comunicação via Wi-Fi foi utilizada para substituir o leitor RFID para identificar as caixas além de ser utilizada para enviar informações do robô em tempo real para depuração de erros.

Palavras-chave: Competição robótica, Encoders, Robótica móvel.

Abstract

Robotic competitions are growing every year, bringing benefits such as solutions to day-to-day problems, helping in education and increasing the interest of children and adolescents in technological areas. This work focuses on the Robot@Factory Lite competition, presenting improvements for the robot used in the competition. After participating in the first edition of the competition, some defects in the functioning of the robot were found that hindered the performance of the teams, since all teams chose to use the same robot model. A robot was developed using encoders to replace time-based motion with distance-based motion. The speed control is done through a PID controller. An odometry system was implemented to monitor the robot's position. Communication via Wi-Fi was used to replace the RFID reader to identify the boxes in addition to being used to send information from the robot in real time for debugging errors.

Keywords: Robotic competition, Encoders, Mobile robotics.

Contents

List of Tables

List of Figures

Acronyms

ADC Analog to Digital Converter.

AGV Automated Guided Vehicle.

HIL Hardware in the Loop.

IDE Integrated Development Environment.

PWM Pulse Width Modulation.

R@FL Robot At Factory Lite.

TFSM Timed Finite State Machine.

Chapter 1

Introduction

Educational robotics has been increasingly present in the teaching of STEAM areas and one of the used methods is learning through robotic competitions. It is in this context that the Robot@Factory Lite competition arises. The main objective of this work is to create a new prototype for this competition using quadrature encoders to improve the robot's performance.

1.1 Motivation

In its first edition, in the Portuguese Robotics Open of 2019, the Robot@Factory Lite organization provided all competitors with a robot prototype that meets the competition requirements, in addition to a demo code containing motion functions for the robot [1]. All competitors used this prototype and developed their robot control based on the demo code.

The demo code is based on a Timed Finite State Machine (TFSM), so one of the main conditions for the transition between states is a time variable, that means that the distance traveled by the robot is measured by how much time has passed. This robot does not have a speed closed loop control, so the use of a time variable to control the distance traveled by the robot is not reliable and is very susceptible to errors. Due to these errors, the contestants had to restart their attempts frequently, greatly reducing the number of

successful attempts to complete the challenge within the competition time.

Since the track used in the competition never changes, the lines and intersections can be used to minimize the errors due to the time based odometry, but that does not solve all the problems, since there are no intersections to be detected near the boxes the robot has to pick up, the intersections can be misdetected and the speed of the robot will be reduced if the battery voltage level drops, increasing the errors in the distance traveled by the robot.

1.2 Objectives

The main objective of this work is to develop a new robot using encoders as an alternative to be used in the competition and also present new features to help the competitors solve the challenge in a faster and more efficiency way, in order to achieve this some secondary objectives are presented:

- Adapt the robot architecture and coding to support the new features;
- Implement odometry based on encoders;
- Implement a velocity PID controller;
- Implementation of Wi-Fi to replace the RFID reader;

1.3 Document Structure

This work has six chapters and is structure as follows:

Chapter 1 presents the introduction, which contains the motivations and objectives of this work.

Chapter 2 describes a brief state of the art regarding to robot competitions and how it can help solving modern problems and also the problems measuring the effectiveness of the competitors approaches in those competitions.

Chapter 3 describes the first robot prototype and the features available for the competitors and the adopted solutions developed to solve the challenge.

In Chapter 4 the robot with encoders is introduced with all the new features that this new approach brings.

Chapter 5 shows the results obtained in the competition with the old robot and some tests results for the new robot.

Finnaly, Chapter 6 displays this work conclusion.

Chapter 2

Related Work

2.1 Robot Competitions

The challenges presented in robotic competitions provide the opportunity for researchers, students and enthusiasts to come up with creative solutions for day-to-day or factory problems. There are many methods and solutions found over the years, therefore, it is important to have a benchmark of the developed methodologies [2]. Challenges for service robots are made in RoboCup@Home based on household activities. The complexity and performance of tasks have been influencing the challenges of domestic robots for over seven years [3]. By comparing the performance of the participating teams, [4] demonstrates the difficulty in judging the approaches of the competitors during the competition. Because maritime robots can have many sizes, shapes and application solutions, the measurement during the challenges of the euRathlon 2015 competition was based on references from previous editions.

A platform for navigation, control, and localization of Automated Guided Vehicle (AGV) robots is demostrated by [5], based on the challenge proposed in the Robot@Factory competition as a test. The RoCKIn@Work competition challenges the competitors to optimize small and medium factory processes to encourage the development of human-robot cooperation applications [6]. Since the challenge is to simulate a real shop-floor situation,

the developed system is able to avoid obstacles, determine the position of the mobile robot and indicate the paths it must take to get the product to be processed.

By comparing Student Autonomous Underwater Vehicles Challenge - Europe (SAUC-E), An Outdoor Robotics Challenge for Land, Sea and Air (EURATHLON) in the 2014 and 2015 editions, and The European Robotics League (ERL) EMERGENCY 2017, [7] demonstrates the instability in the scoring and judging system of the teams during the challenges. While some systems demonstrate to favor the implementation of the application, other systems favor the development of the application project. On the other hand, participants are free to undertake any approaches as long as they respect the rules of each competition.

The robotic competitions present standard problems that can be used as a benchmark, in order to evaluate and to compare the performances of different approaches. Although there are many robotic competitions, such as RoboCup-Industrial, RoboCup Logistic League or RoboCup@work, in order to solve new challenges, there is a necessity to create new ones [8]–[12]. The robot prototyping and programming, in a competition context, can play an important role in education due to its inherent multidisciplinary approach [13], which can motivate students to bridge different technological areas as well as the development of leadership skills and teamwork. It can also play an important role in research and development because it is expected that its outcomes will later be transferred to real-world problems in manufacturing or service robots. The presented work is focused in the Robot@Factory Lite competition that focus on students in the last years of the secondary school. Nevertheless, it can be useful for the first years of the university studies.

2.2 Odometry

Odometry is the measurement of the distance traveled by an object and it is a fundamental method used by robots for navigation. It is easy to measure time in a embedded computer but the speed is usually measured indirectly through the power of the motors, but this relation can be nonlinear and vary with time. Even with wheels with encoders, that makes

the measurement of speed more reliable, the wheels can slip and skid, so there will be an error relating the motion of the wheels to the motion of the robot[14].

2.2.1 Encoders

An encoder translates the turn of the wheels into the corresponding traveled distance, usually by generating certain number of ticks or pulses per rotation of the wheel. For a robot with two wheels, each connect to a motor through a gearing, Given D as diameter of the wheel, R as the rosolution of the built-in encoder in the motors and G as the gear ratio between the motor and the wheel, it is possible to obtain the conversion factor F in equation 2.1, that is the distance traveled by the wheel for each encoder pulse count[15].

$$F = (D * \pi)/(G * R) \tag{2.1}$$

It is also demonstrated by [15] that the traveled distance of each wheel is $T = I * F$, where I is the encoder count, and the center point of the robot the traveled distance is shown in equation 2.2

$$T_c = (T_r + T_l)/2 \tag{2.2}$$

Equation 2.3 shows how to calculate the change in orientation ΔO, B is the distance between wheels.

$$\Delta O = (T_r - T_l)/B \tag{2.3}$$

If the robot is able to make turns in 90 degrees and truly go straight, trigonometry is not necessary in the calculation of the coordinates X and Y, it can be calculate only by increments or decrements of the traveled distance T_c.

Chapter 3

Robot@Factory Lite First Approach

The Robot@Factory Lite is a new competition based on the Robot@Factory competition. This competition presents a problem inspired on the deployment of autonomous mobile robots on a factory shop floor. The robot should be able to transport materials between warehouses or machines that process those materials. In the first round, the robot should move the boxes between the incoming and outgoing warehouses. In the second round, two of the four boxes must be processed by a machine and in the third round some boxes must be processed by both machines. At the start of every attempt the boxes will be randomly placed in the incoming warehouse. The goal is to transport as many boxes as possible in the shortest time. The competitors can make as many attempts within 10 minutes. The different boxes can be recognized by the robot through a RFID reader.

There is a robot model given by the organization but the competitors can use any robot that is within the competition allowed size. The robot must be completely autonomous and cannot establish any kind of communication with an external system that is not explicitly provided by the organization.

The competition track simulates a factory floor and is shown in 3.1, it has a maximum dimension of 1.7 x 1.2 m.

For the next edition, the competition will provide a task assignment server that can replace the RFID reader. It is Wi-Fi server that informs the robot about the parts types that will be available after the start of each run. The robot must send an UDP packet to

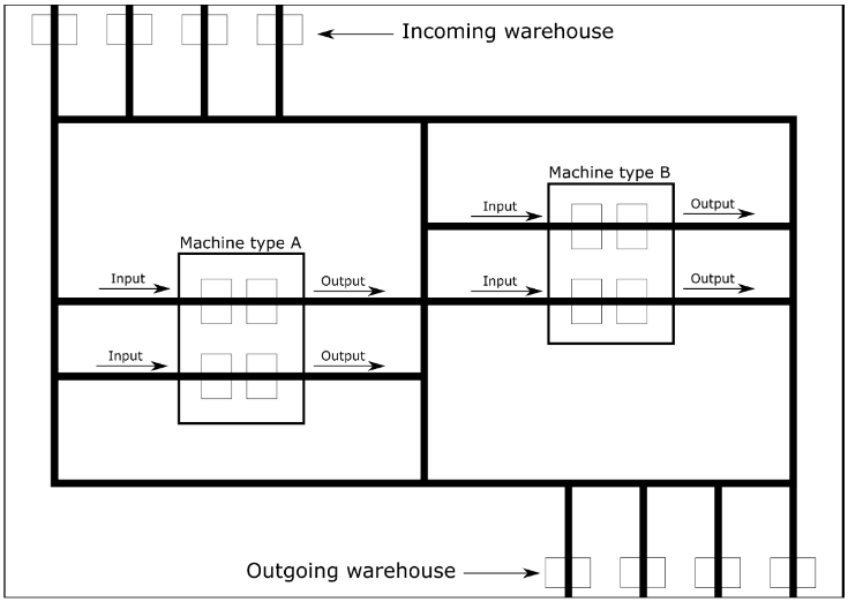

Figure 3.1: Competition Track[1]

the server asking the incoming warehouse parts information with the string "IWP". The server will send an UDP packet to the robot with the part information. The robot can repeat this request throughout the attempt.

3.1 The Robot

The robot used for this competition is part of a kit designed by the competition organizers and it can be found in [1]. A list of all the key components used to built the robot are listed in Table 3.1.

There are some unlisted materials necessary to the assemble of the robot as the 3D printed parts, connection wires and screws.

Since the batteries have to be charged regularly there is a risk of placing the batteries in the wrong electrical polarity when plugging in the robot again. For that purpose it was implemented the circuit in Figure 3.2 for low power losses. The MOSFET used in this circuit was the FDP3682.

The electric schematic with all the connections are shown in Figure 3.3. Note that the

Table 3.1: List of Components

Components	Quantities
DC Motor	2
Wheels	2
RFID Reader	1
Line Sensor	1
Electromagnet	1
Caster Wheel	1
Battery support	1
Motors driver	1
Arduino Uno	1
Arduino Uno expansion shield	1
Lithium battery	2
Step down converter	1
Micro switch	1
Power switch	1
FDP3682	1

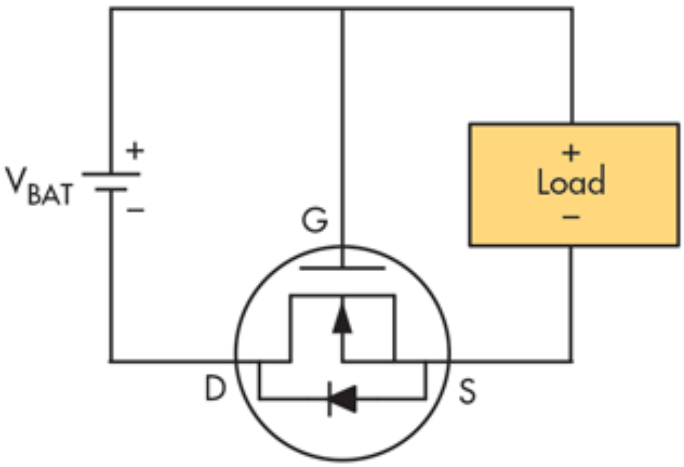

Figure 3.2: Reverse Polarity Protection[16]

digital pin 11 is being used both for the microswitch and the MOSI RFID pin, since there is no more pins available. For that to work properly a 1 kΩ resistor is used in series in the microswitch pin so the digital pin will not be grounded. Without the resistor whenever the microswitch is active we would not be able to communicate with the RFID Reader.

Another approach that can be implemented is to use a dual way motor driver that needs only 4 pins to control two DC motors, this way two digital pins can be used for other purposes.

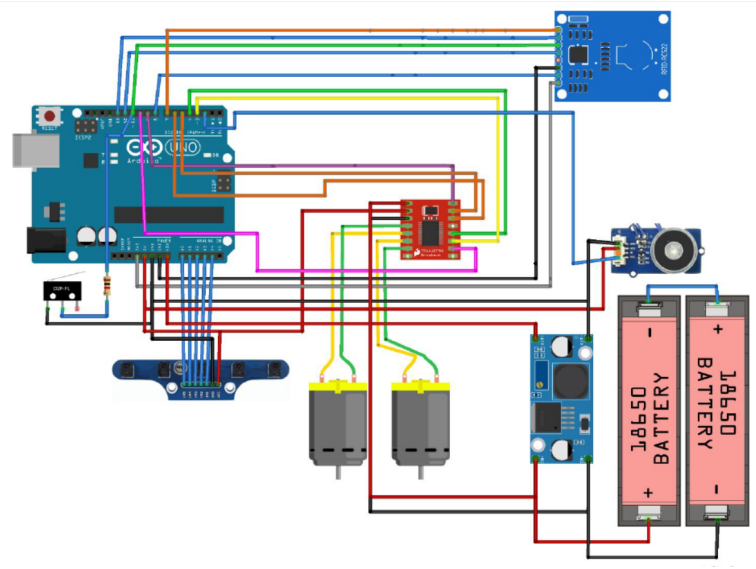

Figure 3.3: Electric Schematic

The assembled robot is showed in Figure 3.4.

3.2 Robot Features

As part of the robot kit, there is a demonstration code of the robot picking up the first box and placing it in the outgoing warehouse [17]. So the code provides some functions that are ready to be used by the competitors in their attempt to solve the challenge.

3.2.1 Robot Movement

There are four movement functions ready for use, where three of them are by following a line either by the right edge, the left edge or by both of them and the last function just moves the robot with a linear and angular speed. The functions are shown in Listing 3.1.

Listing 3.1: Robot movement functions

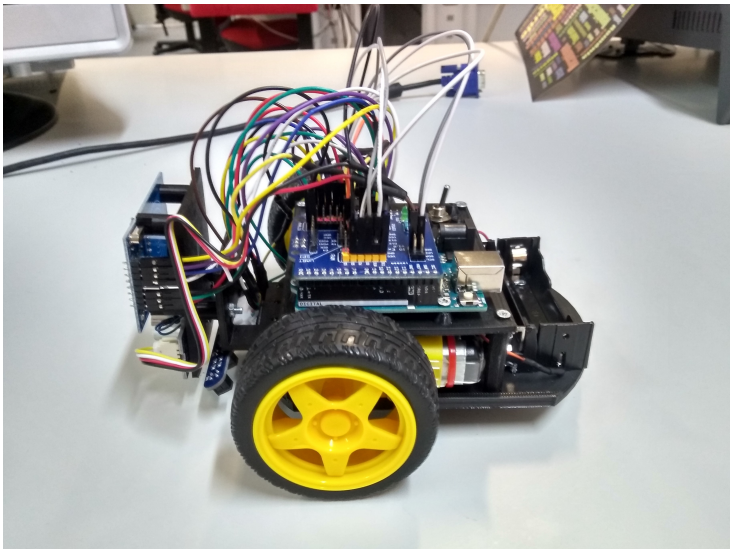

Figure 3.4: Assembled robot

```
void followLineRight(float Vnom, float K)
{
  robot.v = Vnom;
  robot.w = K * IRLine.pos_right;
}

void followLineLeft(float Vnom, float K)
{
  robot.v = Vnom;
  robot.w = K * IRLine.pos_left;
}

void followLine(float Vnom, float K)
{
  float pos;
  if (fabs(IRLine.pos_left) < fabs(IRLine.pos_right)) {
    pos = IRLine.pos_left;
  } else {
    pos = IRLine.pos_right;
  }
  robot.v = Vnom;
  robot.w = K * pos;
}

void moveRobot(float Vnom, float Wnom)
{
  robot.v = Vnom;
  robot.w = Wnom;
}

void loop(){
...
setMotorsVoltage(robot.v + robot.w, robot.v - robot.w);
...
}
```

Note that speed values assigned for each wheel are just Pulse Width Modulation (PWM) values and do not correspond to the actual wheel speed, since this robot does not has a closed loop speed control. The *IRLine.pos* values are calculated in the *IRLine.cpp* code available in [17].

In the example the robot makes a turn using the follow line function, in order to make the robot movement more precise a turn function was develop as shown in Listing 3.2. The function consist of a specific case in the *moveRobot*() function where the linear speed is zero.

Listing 3.2: Turn functions

```
void turn_right(int w)
{
  robot.v = 0;
  robot.w = -w;
}

void turn_left(int w)
{
  robot.v = 0;
  robot.w = w;
}
```

3.2.2 Hardware in the Loop

Along with the prototype kit, the organization also provides to the competitors a simulation model. The SimTwo simulator is an open source software that uses XML language for the graphic part and Pascal language to run the simulation script and those elements can be freely modified, so the teams can adapt their robot model if they made a different one, or modify the script to validate their approach for the competition. The simulation environment can be seen in Figure 3.5.

The Figure 3.6 shows the 3D model of the AGV. The red cylinders represents the five outputs of the line sensor, the green cylinder represents the electromagnet and, finally, the grey rectangular prism represents the RFID reader.

Although the simulation is a good way of validate a solution for the problem before

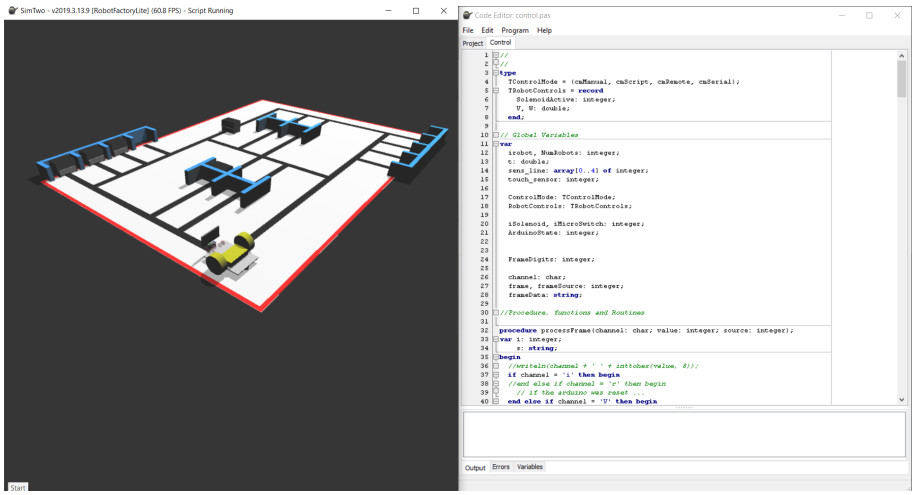

Figure 3.5: SimTwo. Left window is the graphic environment and right window is the code editor[18].

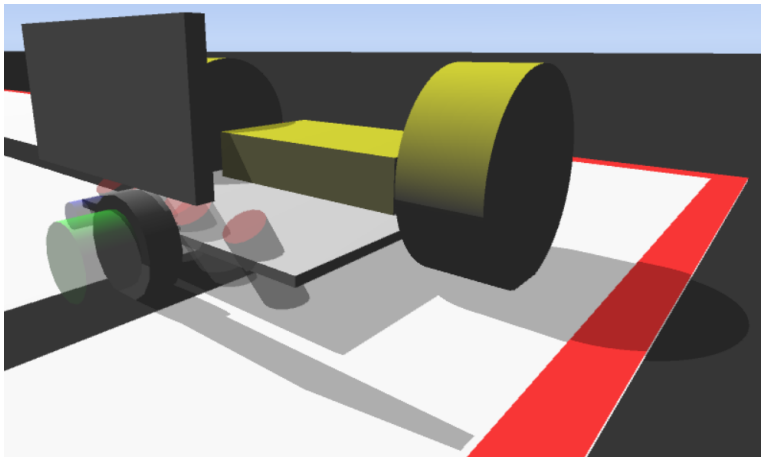

Figure 3.6: AGV simulation model

taking it to the real scenario, the simulation script runs in a different programming language than the proposed robot kit, that uses an Arduino microcontroller. This can make the use of the simulation unproductive, since it would require to adapt the code from one language to another. Also, the simulator is realistic, but it does not consider the microcontroller limitation, such as processing speed and available memory.

In this way, a Hardware in the Loop (HIL) tool is provided by the competition staff [19], so the competitors can insert their microcontroller in the simulation. The way this tool works is, the competitors program their solution to their microcontroller and then, using

a function available in the demo code, the microcontroller is inserted to the simulation loop by serial communication (USB). In this sense, the simulator sends the sensor data (line sensor, RFID and the micro switch) to the microcontroller and the information is processed. Soon after, the microcontroller sends the motors speeds and the electromagnet state to the simulator which is then processed dynamically and graphically.

Using this tool all the processing is made in the microcontroller and the simulator only provides the sensors data and a visual feedback, taking in consideration the real microcontroller limitations. The simulator runs roughly at the same time as the microcontroller loop, which in our case the main loop runs every 40 ms due to limitations in the SimTwo simulator. It also makes possible to use the same code for the simulation and for the real environment. Figure 3.7 demonstrates this program loop.

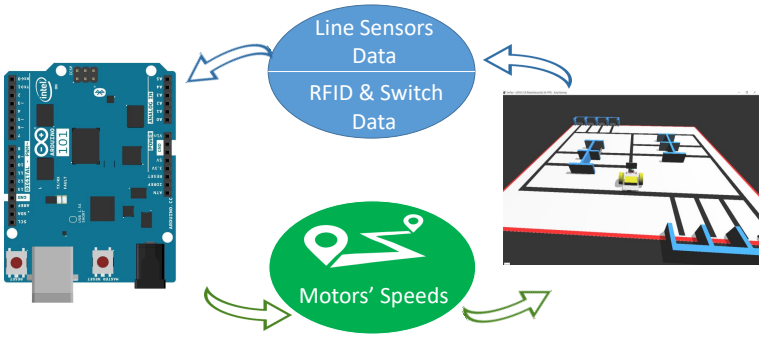

Figure 3.7: HIL illustration [18]

3.3 Adopted Solutions

As stated in the official rules [1], the competition has three rounds, in which new materials are introduced to the challenge as each round advances. The boxes, that represents the materials, were identified through an RFID tag to differentiate the product type that they contain. The following subsections shows the logic of the code developed by our team IPB@Factory for the competition and the states that the robot perform through illustrative figures.

3.3.1 First Case

For the first round, it was used a TFSM technique, following the demo code as an example. Each state represents an action for the robot and the state transition is made by time and other variables, such as the micro switch state and the detection of how many lines the robot passed by. Figure 3.8 demonstrates the pick-and-place process of the first box and figure 3.9 shows the TFSM of this process. The numbers on the first figure are just to represent the robot movements and does not corresponds to the state number.

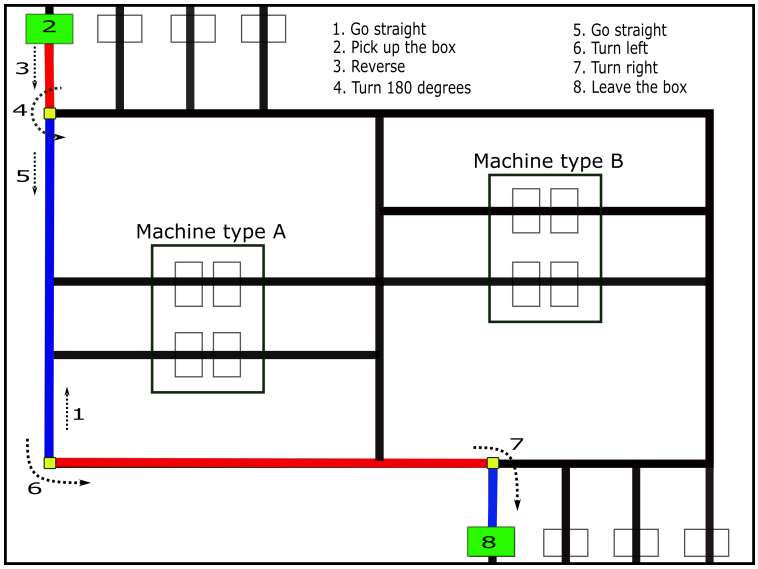

Figure 3.8: Example of a path to deliver the first box [18]

The robot is stopped in state one until the touch switch is activated, in state 2 the robot follows the line by the left edge until the touch switch is activated, indicating it reached the box, then turns the solenoid on (state 3) to grab the box and starts moving backwards in state four for 1.2 seconds. After that the robot turns right for 1.6 seconds, making a 180 degrees turn and transitions to state six. After detecting three crossed lines the robot goes into state seven, making a turn to the left for 0.6 seconds. Then, goes straight until two intersections are detected and makes a turn to the right in state nine. Finally, the robot goes forward in state 10, turns off the solenoid and goes backwards in state 11 and stay stopped in state 12. The state machine is similar for the other boxes.

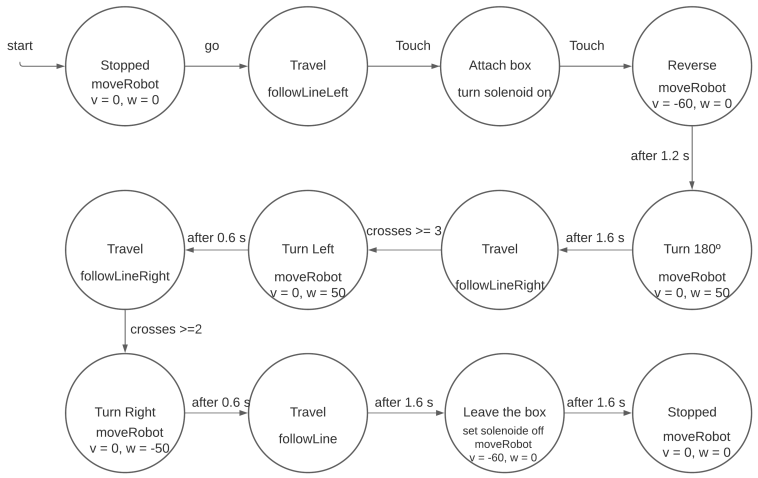

Figure 3.9: TFSM for the first box

3.3.2 Second Case

The second round includes the semi-processed materials, that must be processed by either one of the two machines. With this addition there are twice as more possible paths that the robot could have to perform, that means a large addition to the code using the same approach as the first round, which could result in a lack of memory in the Arduino.

It was noticed that several paths were repeated in the code with minimal changes. For example, the path to go from the incoming warehouse to the outgoing warehouse is almost the same for all situations, as can be seen in Figure 3.10 and 3.11. So, to shrink the code a generic path-travel function was created, where the relevant information, such as velocity and which trajectory the robot should make, were transmitted by parameters.

3.3.3 Third Case

In the third round of the competition, the raw material boxes were included. These boxes should be collected in the incoming warehouse, then be processed by Machine type A, after by Machine type B and finally delivered to the outgoing warehouse.

The algorithm used for this round is the same as the second round, the difference being that new paths were added to the generic path-travel function, since those paths to

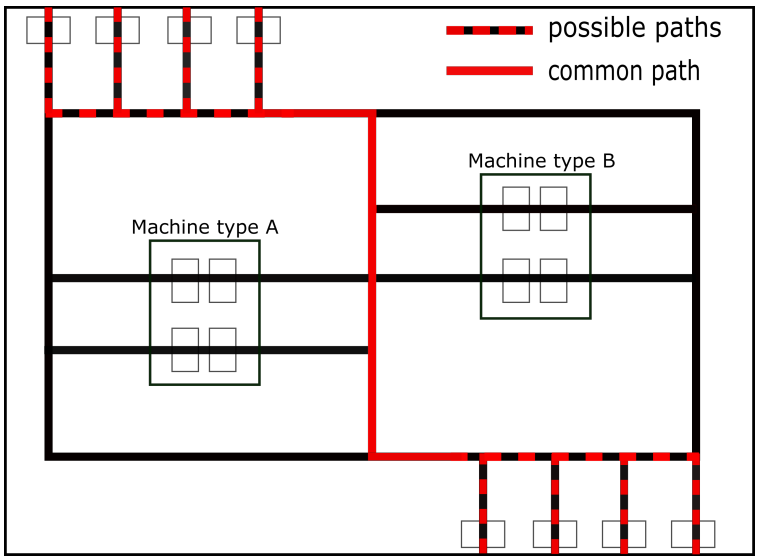

Figure 3.10: Example of a path to deliver the processed parts [18]

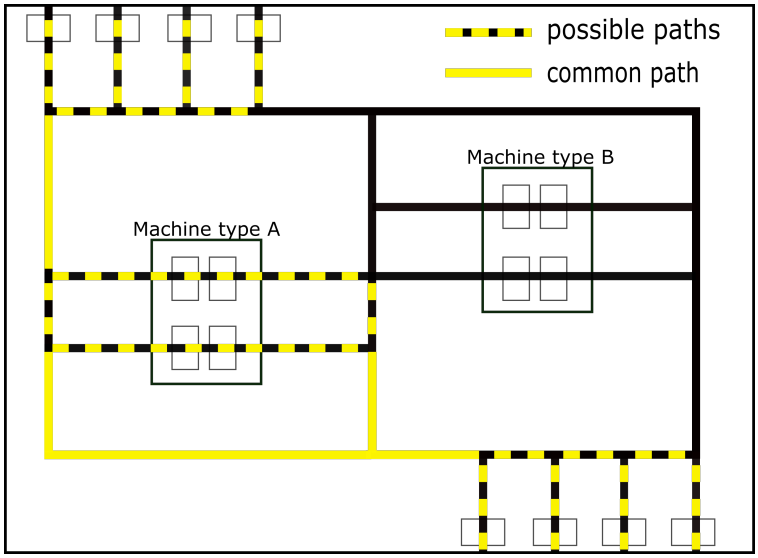

Figure 3.11: Example of a path to the semi-processed parts [18]

the other machine were not used in the other round, and the robot has now three possible tasks for each box. Figure 3.12 shows the new paths add to the function.

Since the robot only knows the type of the material in the moment it picks it up, the algorithm has to be ready to handle all possible material types variation in all the incoming warehouses. This means that the robot is ready to handle all four boxes with raw material, even if it is an impossible scenario by the rules.

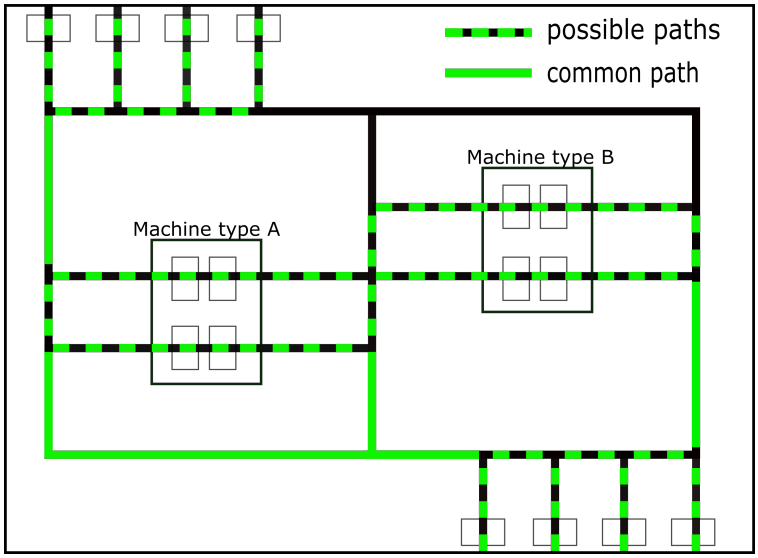

Figure 3.12: Example of a path to the raw parts [18]

The developed algorithm is ready to handle all position probabilities of all types of boxes for all three rounds, so it is possible to use the same code in any round without making any changes in it.

Chapter 4

Robot Improvement

The major flaw in the robot build presented in the last chapter is that it is highly inconsistent, specially when it has to travel long paths, as it is the case in the third round of the competition, where all the competitors had to restart their runs very often, which led to none of the teams being able to carry all four boxes in the last round, within the ten minutes, in the 2019 edition of the competition.

The robot often loses the tracking of it's path due to a misdetection of a crossing line or to an early transition of state. This last problem occurs mainly by two reasons, the first one being losing traction with the ground for a moment, so the robot thinks it traveled the correct distance when it actually did not, and the second being that the battery voltage is lower than it was supposed to be and the robot travels a shorter distance, because the speed control is made by an open loop, where the target speed is a constant PWM value that is set by the user in every state transition.

So to solve some of these problems this chapter presents the use of DC motors with incremental encoders as a solution.

4.1 Hardware Changes

In order to add two encoders in the prototype it is necessary four digital pins to read them, but it was shown in section 3.1 that there is no available pins to be used.

In the next edition of the Robot At Factory Lite (R@FL) competition a Wi-Fi server will be available to inform the robot about the parts [1], so the RFID reader, that uses five digital pins, can be removed. That allows the use of four digital pins to read the encoders and the micro switch does not have to share a digital pin anymore.

Since the Arduino Uno board that was being used does not have Wi-Fi communication it has to be replaced. The new board should be able to run the previous code as less modifications as possible. With that in mind, it was chosen the ESPDUINO-32 board from Wemos, it is compatible with the Arduino IDE, has the same pinout and dimensions as the Aduino Uno board and it uses an ESP32-WROOM-32 microcontroller from Espressif Systems. Figure 4.1 shows a representation of this board.

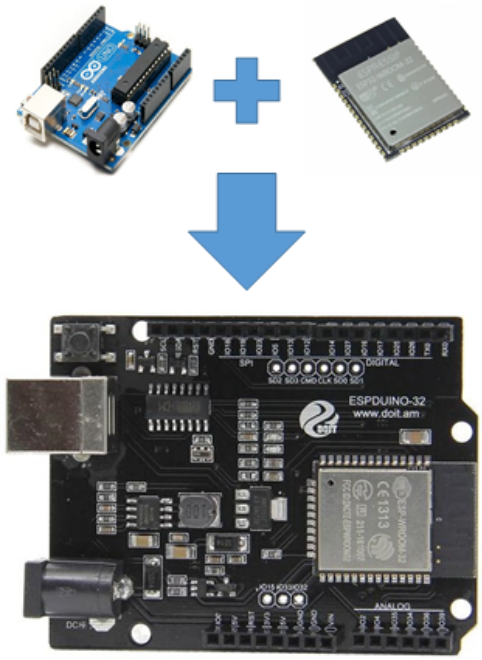

Figure 4.1: ESP32 microcontroller assembled in Uno shield (ESPDUINO-32) [20].

The new connections diagram is shown in Figure 4.2. The line sensors and the electromagnet are not shown in this diagram, but they remain the same, as presented in Figure 3.3.

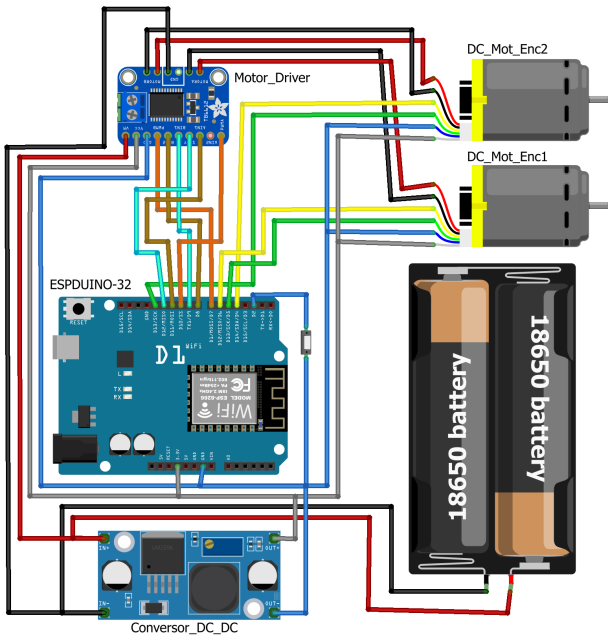

Figure 4.2: New hardware connections diagram.

4.2 Software Changes

Besides the new features that were inserted in the new version, that will be detailed in the next section, since the new board is compatible with arduino IDE, there are just a few changes in the code configuration.

4.2.1 PWM

The PWM configuration is different, for an Arduino board you only have to use the *pinMode*() function to set the pin as an output and then use the *analogWrite*() function to set the PWM duty cycle. For the ESPDUINO32 it is necessary to use two different function to configure the PWM and another one to set the duty cycle, as shown in Listing 4.1.

Listing 4.1: PWM configuration example

```
ledcSetup(PWM_Channel, PWM_freq, PWM_resolution);
ledcAttachPin(MOTOR_PWMA, PWM_Channel);
ledcWrite(PWM_Channel, c);
```

The first function configures a PWM channel to the desired frequency and resolution, which in this case the chosen values are 5 kHz for the frequency and 8 bits for the resolution, to match the old version resolution.

The second function attach a pin to the configured channel and the third function sets the duty cycle of the PWM, where c is a variable between 0 and 255.

4.2.2 Loop function

The last version ran the *void loop*() function every 40 ms, since the new board has a much higher processing clock, that value was changed to 1 ms to allow a faster control.

4.2.3 ADC

The ESPDUINO32 board has two internal Analog to Digital Converter (ADC) but the ADC2 should not be used together with the Wi-Fi, or it can cause conflicts as stated by the manufacturer. Also, since the ADC resolution is 12 bits it had to be configured as a 10 bits resolution in order to use the same follow line function as the old version.

4.2.4 Port Numbers

The last change in the software configuration is in the port numbers, since the nomenclature of the ports are completely different between the two boards. Table 4.1 shows all the ports number changes.

Table 4.1: Ports Changes

	Pins of Hardware	Ports	
		Old Port (Uno)	New Port (ESPDUINO)
Motor Driver	AIN1	6	23
	AIN2	5	19
	PWMA	9	5
	BIN1	4	13
	BIN2	3	12
	PWMB	10	14
Electromagnet	Sol_In	2	25
Box Sensor	Touch_Switch	11	26
Encoders	EncA_ChA	-	27
	EncA_ChB	-	16
	EncB_ChA	-	17
	EncB_ChB	-	18
Line Sensor	IR5	A0	32
	IR4	A1	39
	IR3	A2	36
	IR2	A3	34
	IR1	A4	35

4.3 New Features

Along with the adaptations for the new architecture, there are some new features that were not present in the previous approach. These new features are presented bellow.

4.3.1 Encoders

The DC motor model FIT0450 was used for this new approach (Figure 4.3). The integrated quadrature encoder is placed after a 120:1 gearbox reduction and it has a resolution of 16 simple pulses per round, that provides an output of 1920 pulses in one rotation of the wheel, 960 pulses for each channel.

The pulses generated by the encoder are actually rising and falling edges of a square ware, generated by four magnets when passing through a hall sensor, this gives us four

rising edges and four falling edges for each channel. The frequency of this square wave is proportional to the speed of the motor.

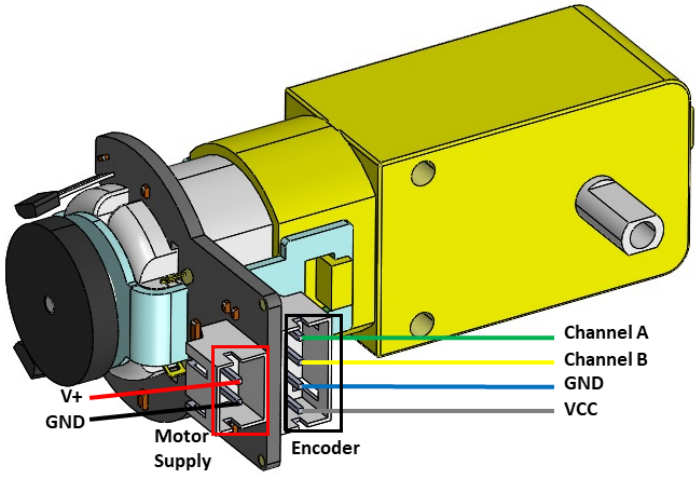

Figure 4.3: DC motor model FIT0450 with encoder. Image adapted from [21].

Figure 4.4 shows the encoder outputs, channel A in yellow and channel B in blue, with the DC motor at full speed forward.

Transition detection

To detect all the transitions in the encoder outputs the built-in timer was set to trigger an interruption every 50 μs, that is enough time to make a read between the falling edge of channel A and the rising edge of channel B, at the highest frequency this signal can reach.

The counting of the pulses is by the detection of a falling edge on channel A, so if at that moment channel B is at a low value the pulse count increases and if it is at a high value the pulse count decreases. The logic is inverted for channel B for the other wheel, since the motors are mirrored in the robot. The code for this logic is shown in Listings 4.2 and 4.3 where the *encA* and *encB* are the pulse counts variables.

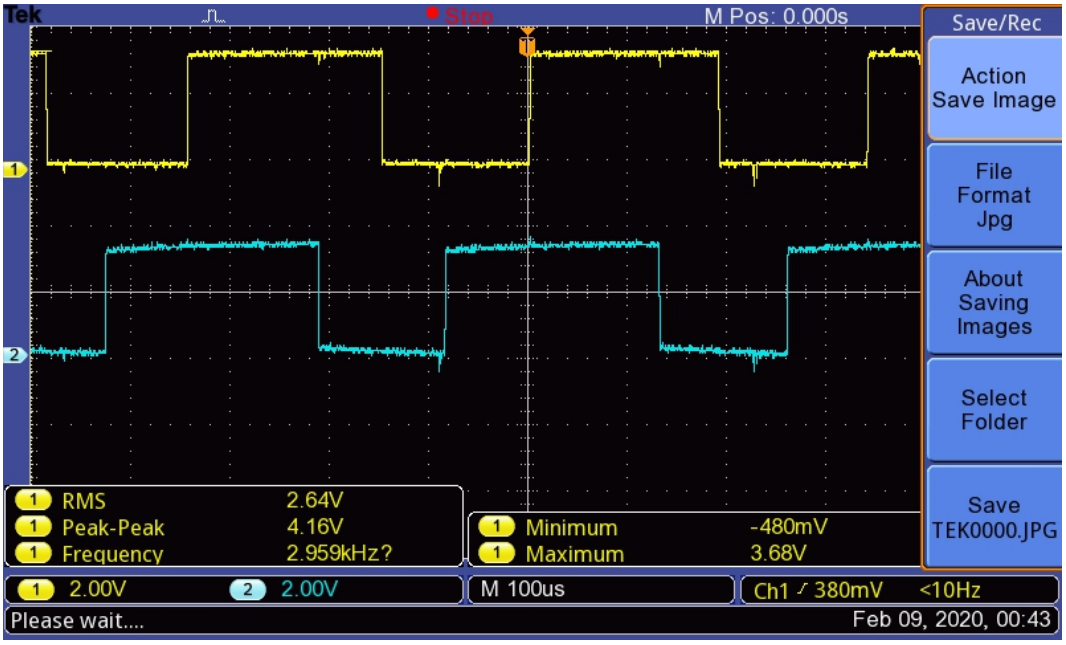

Figure 4.4: Encoder outputs at full speed

Listing 4.2: Right wheel pulse detection

```
encApinARead = digitalRead(encApinA);
if(encApinARead==HIGH && encApinALast==LOW){
  encApinBRead= digitalRead(encApinB);
  if(encApinBRead == HIGH){
  encA++;
  }else{
    encA--;
  }
}
encApinALast = encApinARead;
```

Listing 4.3: Left wheel pulse detection

```
encBpinARead = digitalRead(encBpinA);
if(encBpinARead==HIGH && encBpinALast==LOW){
  encBpinBRead = digitalRead(encBpinB);
  if(encBpinBRead == LOW){
  encB++;
  }else{
    encB--;
  }
}
encBpinALast = encBpinARead;
```

Distance calculation

The ratio of the wheel circumference by the number of pulses per revolution gives us the distance traveled by the wheel per single pulse, the wheel used for this robot with 66 mm of diameter gives us a distance traveled of 0.216 mm/pulse, that is our theoretical maximum error.

Knowing the number of pulses generated by the encoders, now we are able to calculate the linear distance traveled by each wheel. Listing 4.4 shows the proposed function to calculate the estimated velocity for each wheel ($v1e$ and $v2e$), the linear speed ve and the

angular speed we, ds and $dtheta$ are the cumulative traveled distance and angle of the robot. $r1$ and $r2$ are the wheels radius and b is the distance between the wheels.

Listing 4.4: Distance, Velocity and Angle calculations based on odometry function

```
typedef struct{
  int enc1, enc2;
  float v1e, v2e;
  float ve, we;
  float ds, dtheta;
  float r1, r2, b;
  ...
} robot_t;

R.b = 0.071;
R.r1 = 0.033;
R.r2 = 0.033;

void odometry(robot_t& R)
{
  float dt = 1e-6 * interval;

  R.v1e = TWO_PI * R.r1 / (960 * dt) * R.enc1;
  R.v2e = TWO_PI * R.r2 / (960) * dt) * R.enc2;

  R.ve =  (R.v1e + R.v2e) / 2;
  R.we =  (R.v1e - R.v2e) / (2 * R.b);

  R.ds += R.ve * dt;
  R.dtheta += R.we * dt;
}
```

It is important to notice that the 1920 pulses per revolution are from both channels, that is why we have to divide this value by two to get the real detectable number of pulses per revolution.

4.3.2 PID

With the measurement of the speed it is possible to implement a closed loop control with a PID controller. To execute that it was used the PID arduino library[22]. The PID tuning parameter are $Kp = 1.0$, $Ki = 5$ and $Kd = 0.01$, with a sampling time of 40 ms. These values were manually tuned until reaching a stable and fast response. Figure 4.5 shows the speed response when the setpoint is set to 100 mm/s for 15 seconds, than to 0 mm/s for 15 seconds and finally to -100 mm/s for 15 seconds.

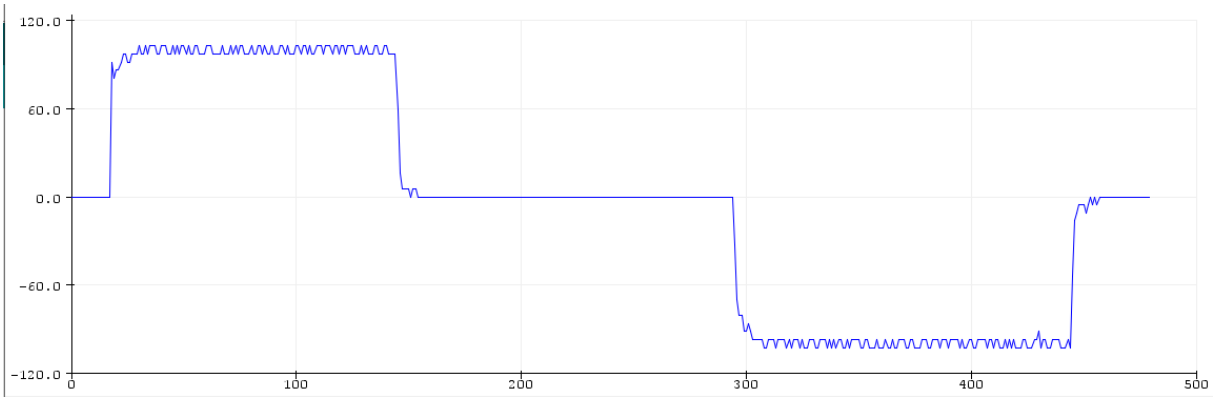

Figure 4.5: Speed response

Gradual speed reduction

Even with a speed response close to a square, due to the inertia, the robot stills moves for a fraction of time after it is supposed to stop, this time is proportional to the speed it was before stopping.

To avoid this behavior that decreases the robot accuracy it was proposed a function that gradually reduces the robot speed before it reaches the stopping point. Listing 4.5 shows the proposed function, where $Vnom$ is the maximum speed, $dist\_error$ is the distance the robot is from the desired point, $target$ is the desired distance the robot is supposed to travel and Kp_v is the constant that control the speed reduction.

Listing 4.5: Speed reduction function

```
void speed_control(float Vnom){
  dist_error = R.target - R.ds;
  R.v = Kp_v*dist_error;
  if(R.v >= Vnom){
  R.v = Vnom;
  }
```

Basically this function implements a proportional controller for the distance error. Figure 4.6 shows the expected speed values for a $Kp\_v = 4$ and nominal speed at 100 mm/s.

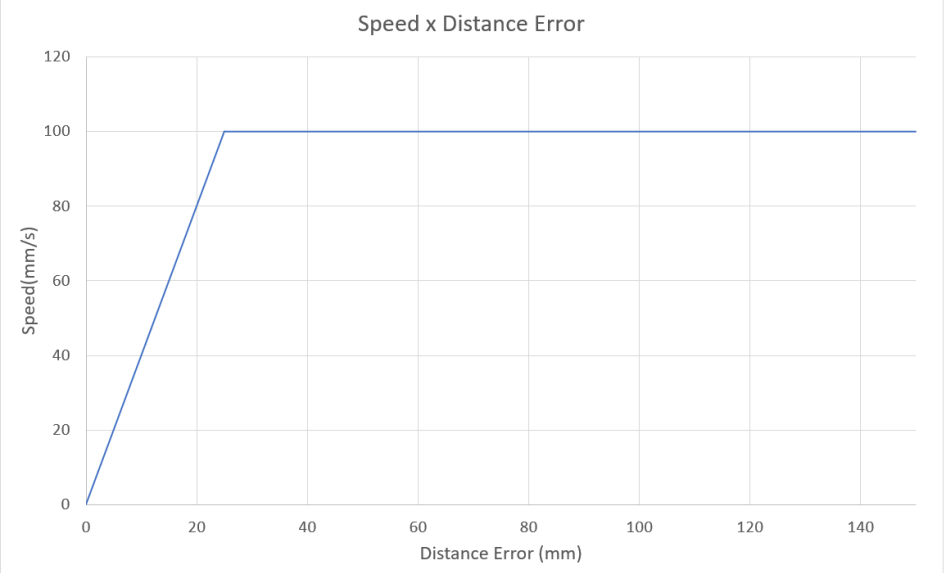

Figure 4.6: Speed reduction

4.3.3 Odometry

Knowing the variations in the distance traveled by the robot it is possible to build a odometry function that allow us to know the coordinates of the robot at any time. Since this competition only requires the robot to walk at one direction at a time (up, down, left or right), the odometry function becomes much more simple. The proposed function is shown in Listing 4.6

Listing 4.6: Odometry function

```
void odometry(int orientation){
  delta_dist = R.ds - last_dist;

  switch(orientation){
    case 0:
    robot.x = robot.x + delta_dist;
    break;
    case 90:
    robot.y = robot.y + delta_dist;
    break;
    case 180:
    robot.x = robot.x - delta_dist;
    break;
    case 270:
    robot.y = robot.y - delta_dist;
    break;
    default:
```

```
    break;
  }
  last_dist = R.ds;
}
```

Note that the *orientation* variable is not *R.dtheta* from Listing 4.4, it is only an approximation for the calculations, not the real robot orientation.

Odometry calibration

Although the odometry for this robot is simple, it can add up a lot of errors trough the tasks, specially if the user makes the robot go forward a little after reaching the box to make sure the robot properly grabs the box. So, since the track always remains the same, to correct these errors a correction function is proposed in Listing 4.7 based on the detection of the track lines.

Listing 4.7: Odometry correction function

```
int matrix_90_y [5][3] = {
  {0,0,0},
  {173,173,322},
  {322,322,471},
  {685,471,685},
  {685,685,685}
};

int matrix_270_y [5][3] = {
  {71,71,71},
  {285,285,434},
  {434,434,583},
  {797,583,797},
  {797,797,797}
};

int matrix_x [3] = {15,755, 1615};

void cross_interrupt(void){

  IRLine.crosses = 0;
  last_x = x;
  if(robot.x < 600){
    x = 0;
  }else if(robot.x > 900){
    x = 2;
  }else{
    x = 1;
  }

  if(last_x == 0 && x == 1 && y == 3)y = 4;
  if(last_x == 2 && x == 1 && y == 3)y = 4;
```

```
  if(last_x == 1 && x == 2 && y == 4)y = 3;
  if(last_x == 1 && x == 0 && y == 4)y = 3;

  if(robot.theta == 90){
    y++;
    robot.y = matrix_90_y[y][x];
    robot.x = matrix_x[x];
  }

  if(robot.theta == 270){
    y--;
    robot.y = matrix_270_y[y][x];
    robot.x = matrix_x[x];
  }
}
```

The intersections can be interpreted as the parameters of a matrix, as shown in Figure 4.7, so every time the robot detects a horizontal intersection, knowing which intersection it is on, it will correct the values for the x and y coordinates. Those values in the matrix are taking in consideration the direction the robot is coming, rather it is from above or from under, and also the distance from the line sensor to the center of the robot.

The first condition in the function changes the row parameter of the matrix based on the x coordinate of the robot.

The second part makes a correction in the line parameter of the matrix because, as shown in Figure 4.7, the center line has five intersection while the side ones have only four. So when the robot travel between them from the top the y parameter changes from 3 to 4 or the other way around, even tough the robot is traveling in the left or right direction.

Finally, the last part checks the robot direction and apply the corrections in the x, y coordinates.

4.3.4 Robot control

The control of the robot remains almost the same as the old version, as a state machine, but now the state transition occurs with the distance traveled by the robot, as shown by the state machine in Figure 4.8.

Listing 4.8 shows the code that implements the first two states of this FSM.

Listing 4.8: Control function

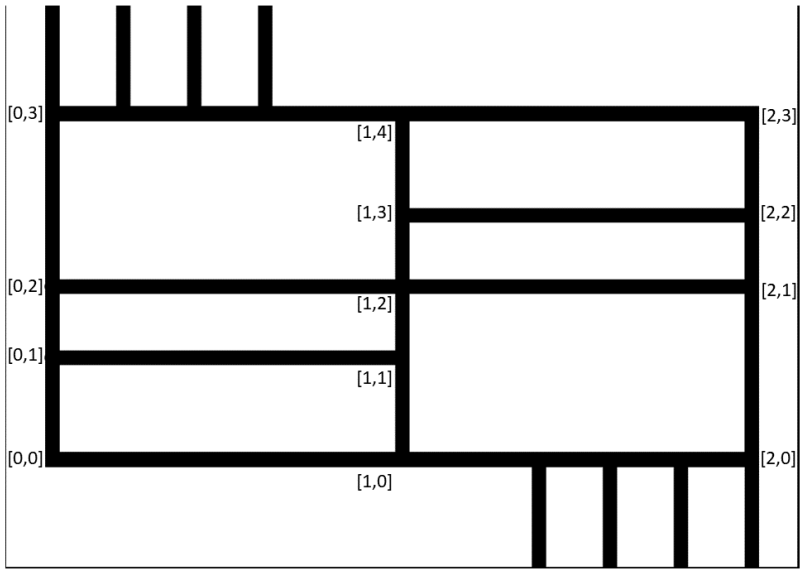

Figure 4.7: Intersections representation

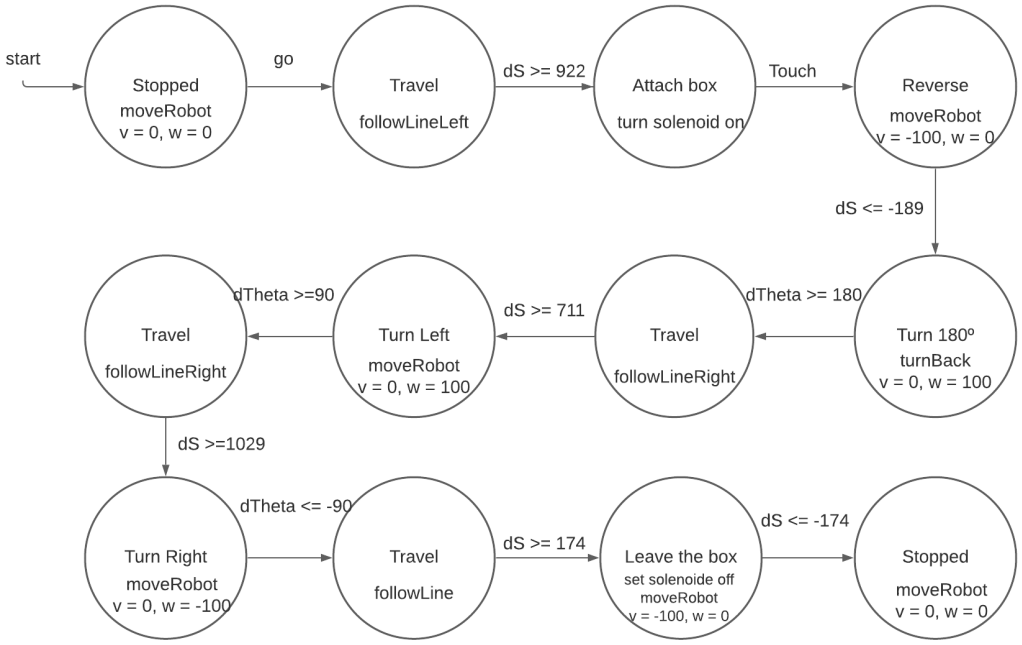

Figure 4.8: Intersections representation

```
void control(void){
  if (robot.state == 0) {          // Robot Stoped
    moveRobot(0, 0);

  } else if (robot.state == 1) {  // Go: Get first box
    followLineLeft(100,-1);
```

```
  }

  if (estado == 0 && LastTouchSwitch && !TouchSwitch){
    estado = 1;
    setAction(1,915+7);

  }else if((estado == 1)&&(abs(dist_error) <= 0.2)&&(TouchSwitch)){
    robot.solenoid_state = 1;
    setAction(2,-189);
    estado = 2;
  }
```

The *setAction*() function first parameter is the next state and the second is the new target distance, in mm. $dist_error$ is the difference between the target distance and how much the robot moved (dS).

4.3.5 Wi-Fi

The Wi-Fi configuration on the ESP32 is really simple using the Wi-Fi library, as shown in Listing 4.9 and it has to be configure before the timer.

Listing 4.9: Wi-Fi

```
#include <WiFi.h>
WiFiUDP Udp;

void wifi_setup(){
while (status != WL_CONNECTED) {
    Serial.print("Attempting␣to␣connect␣to␣SSID:␣");
    Serial.println(ssid);
    status = WiFi.begin(ssid,pass);

    delay(10000);
  }
  Serial.println("Connected␣to␣wifi");
  printWifiStatus();
  Udp.begin(localPort);
}

void printWifiStatus() {
  Serial.print("SSID:␣");
  Serial.println(WiFi.SSID());

  IPAddress ip = WiFi.localIP();
  Serial.print("IP␣Address:␣");
  Serial.println(ip);

  long rssi = WiFi.RSSI();
  Serial.print("signal␣strength␣(RSSI):");
  Serial.print(rssi);
  Serial.println("␣dBm");
}
```

The robot tries to connect to the Wi-Fi network every 10 seconds until the connection is completed. Than, if a serial port is connected, it prints on the serial monitor the name, the IP address and the signal strength of the network. Finnaly it starts the communication with the local port.

To request the box information the following function in Listing 4.10 is proposed.

Listing 4.10: Task request function

```
char packetBuffer[255];
char ReplyBuffer[] = "IWP";
void task_request(){
    Udp.beginPacket(Udp.remoteIP(), Udp.remotePort());
    Udp.printf(ReplyBuffer);
    Udp.endPacket();
}

void loop(){
...

int packetSize = Udp.parsePacket();
  if (packetSize) {
    int len = Udp.read(packetBuffer, 255);
    if (len > 0) {
      packetBuffer[len] = 0;
    }
    Serial.println("Contents:");
    Serial.println(packetBuffer);
  }
...
}
```

The robot sends the request and keep checking in the *loop*() function for the reply from the server.

It is also possible to use the request function to send any data from the robot, for example speed and position, to check the performance of the robot in the middle of the competition or to debug any defect on the code with the robot on-the-fly. The Wi-Fi opens a lot more possibilities than just receiving the box information.

Chapter 5

Results

5.1 First Approach

The first tests consisted of verifying the robot movement functions provided to the competitors such as the follow line, move robot and turn mentioned in section 3.2.1 and set the appropriate settings to adjust the movements of the robot.

The simulation environment was applied in the initial tests by the team members to become familiar with the competition scenario and to develop basic instruction for the robot decisions. The possibility of performing the experiments without the physical robot is a great tool but the simulation is only really usefull in the early stages of code development due to the differences between the real robot and the simulated model, specially in the time values of the state transitions.

After that, the development of the codes to solve the first challenge started. As mentioned in section 3.3.1 we followed the TFSM technique used in the demo code for the first box and expended it for the other boxes. So, to complete the first round, 61 states were needed and 51% of the microcontroller‘s memory (SRAM) was used. The memory usage is estimated by the Arduino Integrated Development Environment (IDE).

The code for the second round was based on a single function that covered all the necessary paths, reducing the size of the code, since paths that were similar now are

executed by the same function. Each path was represented as a switch case statement, so the path can be traveled more than once, using different parameters.

In the main function of the program, exemplified by the pseudocode in Listing 5.1, the detection of the product type is performed through an RFID tag reading function and the execution of the pick-and-place process is made by the Route function, shown in Listing 5.2. The source code consumed 58% of the microcontroller SRAM.

Listing 5.1: Main Function

```
function Main Function
    Go straight until touch the box
    Box  = RFID Result
    if Box  == Processed Part then
        function ROUTE(1)
        end function
        function ROUTE(2)
        end function
    else if Box == Semi-Processed Part then
        function ROUTE(3)
        end function
        function ROUTE(4)
        end function
        function ROUTE(5)
        end function
        function ROUTE(2)
        end function
    end if
end function
```

Listing 5.2: Route Function

```
function ROUTE(int PathNumber)
    switch PathNumber
        case 1
            Pick up the box and leave in outgoing warehouse
        case 2
            Go back to incoming warehouse
        case 3
            Pick up the box and leave in the machine
        case 4
            Pick up the box in the machine
        case 5
            Leave the box in outgoing warehouse
end function
```

For the third round the route function was upgraded with more paths and the main function has a new possible result for the RFID tags, therefore another group of route functions were added as a possible scenario. The final code used 67% of the microcontroller SRAM.

5.1.1 Competition Performance

The competition was divided into three days, one for each round. The developed robot was able to complete all the requirements in the first two rounds and failed to delivered the last box in the third round, summing a total of 11 delivered boxes, receiving the first prize in the senior category, where the second place was able to delivered 8 boxes in total. Table 5.1 shows the performed times and the delivered boxes in each round. Figure 5.1 shows the robot caring the last box to the first machine in the third round of the competition.

Table 5.1: Performed Times in the Competition.

Round	Time	Boxes
1	2:00'57	4 blue boxes
2	3:48'80	2 blue boxes and 2 green boxes
3	2:58'00	1 red box, 1 green box and 1 blue box

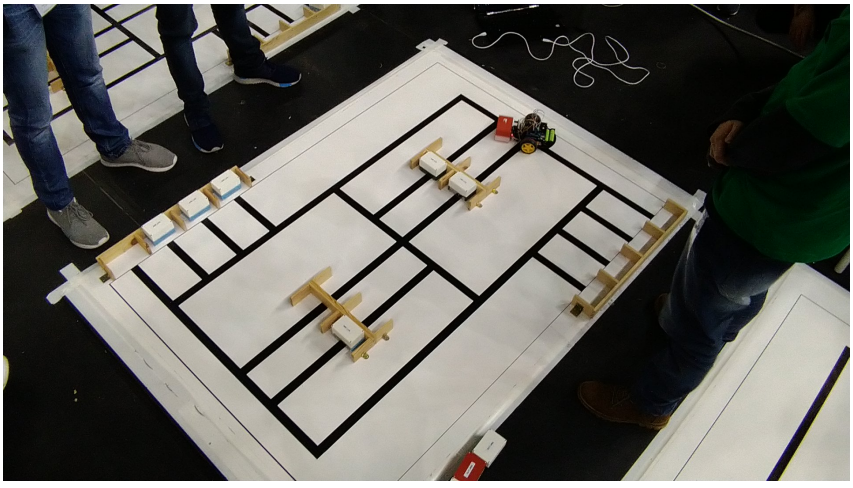

Figure 5.1: Real robot in the third round[18]

5.2 Robot with Encoders

The new robot with encoders does not have a full algorithm to complete the challenge, like the first robot, since the goal for this work it to create new tools and make it available to the competitors for the next competition. So a demo code was developed for the new approach. The video available in [23] shows the robot running the demo code and a program receiving the information it is sending via Wi-Fi every one second.

The robot stars moving at the time 0:09 after activating the touch switch and receive the command to go forward 922 mm with a speed of 200 mm/s, after arriving at the given point and the touch switch gets activated by the box, the robot grabs the box by turning the electromagnet on and goes back 189 mm with a speed of 100 mm/s. Than the robot performs a 180 degrees turn and goes forward 711 mm at 100 mm/s, after that it makes a 90 degrees turn to the left and drives forward 1029 mm at 150 mm/s. Finally the robot makes a turn to the right, goes forward 174 mm, turns off the electromagnet, goes back the same distance and stay parked. Figure 5.2 shows a frame of the video after the robot stops.

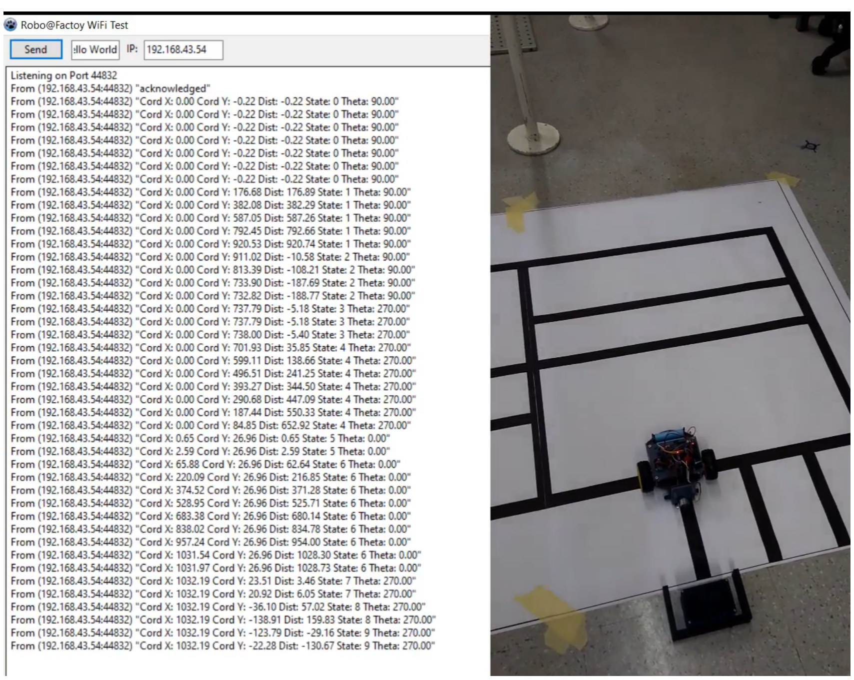

Figure 5.2: New demonstration code test

Since this is a demo code and there is just one box, the robot is not sending requests to the server and is just sending information, like the "x" and "y" coordinates, the distance traveled in the current state, the current state and the robot approximately orientation. This information is customizable by the user and in the competition scenario, the request can be send between this information, since the server will only respond to the request.

In this test the speed reduction constant Kp_v is set at 5, that means that the robot will start gradually reduce the speed 20 mm before the target point.

5.3 Robots Comparison

One of the main advantages with the implementation of Wi-Fi is the capability of the robot to schedule the movements before reading the parts, since this is provided by the referee. Also, it allows the competitors to debug the information the robot is sending in real-time, while in the previous robot, the user has to use a USB cable attached to the robot while it is running to search for flaws in the algorithm.

The use of encoders brings new advantages that could not be reached before, such as a closed-loop speed controller of each wheel and a measure of the distance traveled by the robot. The control of distance for the state transition is now based on encoder pulses instead of time.

This approach brings advantages such as independence of speed in the state transitions, if the speed is changed, either by the competitors to try a faster run in the round, or by the battery charge state delivering less power, the state machine remains the same.

The dependence of the battery was a big factor in consideration when developing the algorithm for the old robot. To expose this and the error that it implies, a simple test was conducted. The robot should go from the official competition starting point, pick up the box and go backwards 0.7 seconds, for the old robot, or go backwards 169 mm for the new robot with encoders. This reference point of 169 mm was the measured distance traveled by the robot without encoders with a voltage supply of 8V. Figure 5.3 shows an illustration of the experiment.

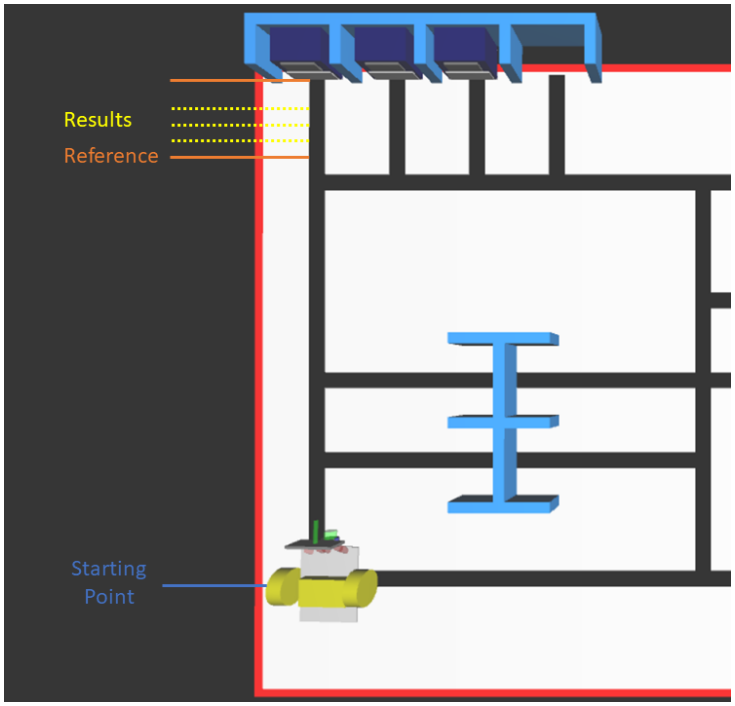

Figure 5.3: Illustration scheme of the experiment[24]

This test was performed with different voltage levels, using an adjustable power supply to simulate the voltage drop in the battery. The tests stated with a 8V power supply and got decrease 0.2V for each cycle until reaches 6V. Therefore, 11 test cycles were performed for each AGV.

At the end of each test cycle the error in the distance traveled by the robot was measured. This error is determined by the distance from the reference point to where the robot ends the path. If the error is positive the robot did not reach the desired point, if negative, the robot passed on that point. Table 5.2 shows the measured distances collected for each voltage applied to each AGV, as well as the error in relation to the reference point.

As show the results the first AGV decreases the distance traveled as the voltage supply gets lower, while the AGV with encoders travels almost the same distance independent of the voltage supply. The maximum error for the first robot is 58 mm while for the new one is just 2 mm.

In a real scenario a 2V decrease in the battery voltage will not happen or will take

Table 5.2: Measured distance by voltage supply for both robots.

	Distance [mm]		**Error [mm]**	
Voltage	**Old AGV**	**New AGV**	**Old AGV**	**New AGV**
8	169	169	0	0
7,8	157,5	168,5	11,5	0,5
7,6	151	168,5	18	0,5
7,4	147	168,5	22	0,5
7,2	140	169,7	29	-0,7
7	135	169,3	34	-0,3
6,8	130	171	39	-2
6,6	125,5	170	43,5	-1
6,4	120	170	49	-1
6,2	115	169,5	54	-0,5
6	111	169,3	58	-0,3

a lot of time, but a 0.1V decrease in each battery, that are in series configuration, can occurs within a few hours of use. The average error increase for every 0.2V decrease in the voltage supply in the old AGV is 5.8 mm that represents an error of 3,43% regarding to the 169 mm goal. So, for a 1 m travel distance, that is common in the competition, this error will represent a 34.3 mm error for a 0.2V battery discharge, that will usually lead the robot to some unexpected behavior.

Figure 5.4 shows a graph of distance by voltage supply for both robots, where the orange line represents the robot with encoders, and the blue line represents the old approach without encoders.

The reason why the distance decrease with the voltage supply in the robot without encoders is that it is in an open loop speed controller, since the duty cycle is a constant when the voltage supply gets decreased the speed also decreases. With a lower speed the robot will travel a shorter distance within the time of the current state.

With the encoders giving feedback of the speed the speed controller will increase the duty cycle to maintain the desired speed. Even if the voltage supply gets low to a point that the duty cycle is at 100% and the speed is lower than it is supposed to be, the robot with encoders would still reach the reference point, because the state machine transitions are based on the distance traveled.

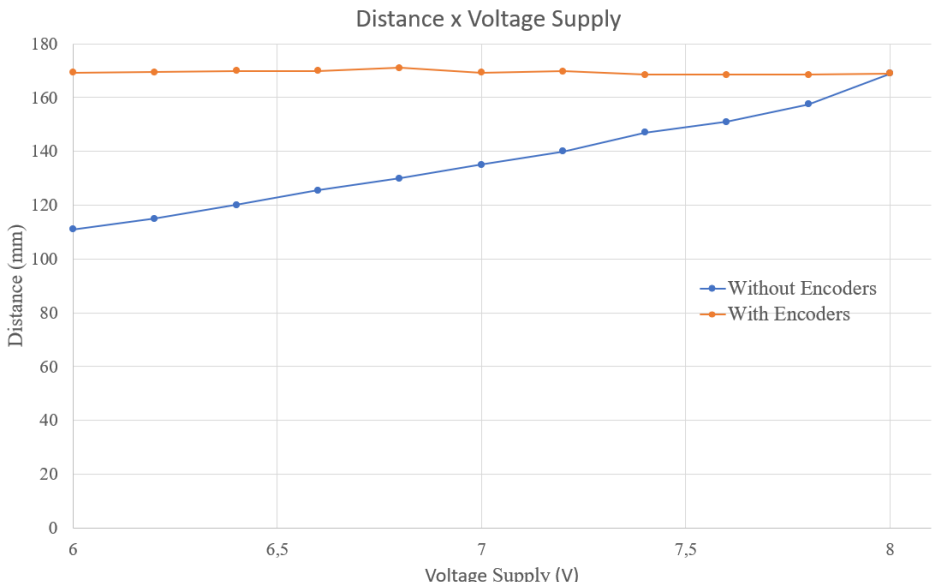

Figure 5.4: Graph of the distance by voltage supply for both AGVs.[24]

Chapter 6

Conclusions and Future Works

6.1 Developed Works

Based on what was presented, a new prototype for the Robot@Factory Lite competition was developed, to create this several developments were made. First, the original AGV prototype was assembled and an algorithm was developed by the IPB@Factory team to complete the challenge. With the collaboration of all members of the team, IPB@Factory managed to win the first edition of the Robot@Factory Lite Competition. Even though great results were achieved in the competitions, several issues were found in the AGV architecture. To solve that a new robot was developed using encoders and a different micro-controller, which made it possible to measure the distance traveled by the robot, control the robot speed with a PID controller, create an odometry function to monitor the robot‘s position, change the state machine transaction from time to distance, making the robot control independent of the battery state of charge, and finally, implement Wi-Fi communication to identify the box types and to get real-time information from the robot. This new robot prototype can be used by the competition organization to offer the competitors new features that allows more precise and efficient solutions for the challenge.

6.2 Future Work

The present work has potential to continue to develop. These will be described below as topics:

- Develop a new adaptive state machine;
- Create a full algorithm to participate in another edition of the competition;
- Study how the robot behave at a higher speed to reach an optimal point between speed and precision of the tasks;
- Implement an encryption in Wi-Fi communication to increase the security of the communication with the referee server;

Bibliography

[1] P. Costa and J. Lima, *Robot at factory lite competition files*. [Online]. Available: https://github.com/P33a/RobotAtFactoryLite.

[2] D. Holz, L. Iocchi, and T. Zant, "Benchmarking intelligent service robots through scientific competitions: The robocup@home approach.," Jan. 2013.

[3] L. Iocchi, D. Holz, J. Ruiz-del-Solar, K. Sugiura, and T. van der Zant, "Robocup@home: Analysis and results of evolving competitions for domestic and service robots," *Artificial Intelligence*, vol. 229, pp. 258–281, 2015, ISSN: 0004-3702. DOI: https://doi.org/10.1016/j.artint.2015.08.002. [Online]. Available: http://www.sciencedirect.com/science/article/pii/S0004370215001174.

[4] Y. Petillot, F. Ferreira, and G. Ferri, "Performance measures to improve evaluation of teams in the eurathlon 2014 sea robotics competition," *IFAC-PapersOnLine*, vol. 48, no. 2, pp. 224–230, 2015, 4th IFAC Workshop onNavigation, Guidance and Controlof Underwater VehiclesNGCUV 2015, ISSN: 2405-8963. DOI: https://doi.org/10.1016/j.ifacol.2015.06.037. [Online]. Available: http://www.sciencedirect.com/science/article/pii/S2405896315002761.

[5] P. J. Costa, N. Moreira, D. Campos, J. Gonçalves, J. Lima, and P. L. Costa, "Localization and navigation of an omnidirectional mobile robot: The robot@factory case study," *IEEE Revista Iberoamericana de Tecnologias del Aprendizaje*, vol. 11, no. 1, pp. 1–9, Feb. 2016, ISSN: 2374-0132. DOI: 10.1109/RITA.2016.2518420.

[6] R. Bischoff, T. Friedrich, G. K. Kraetzschmar, S. Schneider, and N. Hochgeschwender, "Rockin@work: Industrial robot challenge," in *RoCKIn*, Rijeka: IntechOpen,

2017, ch. 4. DOI: 10.5772/intechopen.70014. [Online]. Available: https://doi.org/10.5772/intechopen.70014.

[7] F. Ferreira, G. Ferri, Y. Petillot, X. Liu, M. Franco, M. Matteucci, F. Grau, and A. Winfield, "Scoring robotic competitions: Balancing judging promptness and meaningful performance evaluation," Apr. 2018, pp. 179–185. DOI: 10.1109/ICARSC.2018.8374180.

[8] A. Eguchi, "Robocupjunior for promoting stem education, 21st century skills, and technological advancement through robotics competition," *Robotics and Autonomous Systems*, 2016.

[9] E. Pagello, E. Menegatti, A. Bredenfeld, P. Costa, T. Christaller, A. Jacoff, D. Polani, M. Riedmiller, A. Saffiotti, E. Sklar, and T. Tomoichi, "Robotic competition based education in engineering," *AI Magazine*, 2004.

[10] F. Ribeiro, I. Moutinho, P. Silva, C. Fraga, and N. Pereira, "Controlling omnidirectional wheels of a robocup msl autonomous mobile robot," *Scientific Meeting of the Portuguese Robotics Open*, 2004.

[11] B. Browning, J. Bruce, M. Bowling, and M. Veloso, "Ustp: Skills, tactics and plays for multi-robot control in adversarial environments," *IEEE Journal of Control and Systems Engineering*, 2005.

[12] R. Nakanishi, J. Bruce, K. Murakami, T. Naruse, and M. Veloso, "Cooperative 3-robot passing and shooting in the robocup small size league," *Proceedings of the RoboCup Symposium*, 2006.

[13] E. Sklar, A. Eguchi, and J. Johnson, "Robocupjunior: Learning with educational robotics," *AI Magazine*, vol. 24, no. 2, 2003.

[14] M. Ben-Ari and F. Mondada, "Robotic motion and odometry," in *Elements of Robotics*. Cham: Springer International Publishing, 2018, pp. 63–93, ISBN: 978-3-319-62533-1. DOI: 10.1007/978-3-319-62533-1_5. [Online]. Available: https://doi.org/10.1007/978-3-319-62533-1_5.

[15] "Chapter 13 - knowing where you are," in *Building Robots with LEGO Mindstorms NXT*, M. Ferrari and G. Ferrari, Eds., Burlington: Syngress, 2007, pp. 249–263, ISBN: 978-1-59749-152-5. DOI: https://doi.org/10.1016/B978-159749152-5/50018-8. [Online]. Available: http://www.sciencedirect.com/science/article/pii/B9781597491525500188.

[16] n.p., *Reverse-polarity protection in automotive design*, Power Electronics, Mar. 2016. [Online]. Available: https://www.powerelectronics.com/technologies/power-management/article/21863299/reversepolarity-protection-in-automotive-design.

[17] P. Costa and J. Lima, *Robot at factory lite demo code files.* [Online]. Available: https://github.com/P33a/SimTwo/tree/master/RobotFactoryLite/rafliteduinoHWLoop.

[18] J. Braun, L. A. 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," in *Robot 2019: Fourth Iberian Robotics Conference*, M. F. Silva, J. Luís Lima, L. P. Reis, A. Sanfeliu, and D. Tardioli, Eds., Cham: Springer International Publishing, 2020, pp. 478–489, ISBN: 978-3-030-35990-4.

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

[20] E. Systems, *Arduino core for the esp32*, https://github.com/espressif/arduino-esp32, January, 2020.

[21] DFRobot, *Dfrobot micro dc geared motor with encoder*, https://wiki.dfrobot.com/Micro_DC_Motor_with_Encoder\protect\discretionary{\char\hyphenchar\font}{}{}SJ01_SKU_FIT0450, January, 2020.

[22] B. Beauregard, *Pid library.* [Online]. Available: https://playground.arduino.cc/Code/PIDLibrary/.

[23] V. Oliveira, *New demo code test*, Nov. 2020. [Online]. Available: https://drive.google.com/file/d/1-j5ckIWGSWL09e7UaUa6pqRxUpUrFHe9/view.

[24] J. Lima, V. Oliveira, T. Brito, J. Gonçalves, V. H. Pinto, P. Costa, and C. Torrico, "An industry 4.0 approach for the robot@factory lite competition," in *2020 IEEE International Conference on Autonomous Robot Systems and Competitions (ICARSC)*, 2020, pp. 239–244. DOI: 10.1109/ICARSC49921.2020.9096164.