UNIVERSIDAD DE INVESTIGACIÓN DE TECNOLOGÍA EXPERIMENTAL YACHAY

Escuela de Ciencias Matemáticas y Computacionales

ARTIFICIAL INTELLIGENCE BASED POSITION LOCATOR USING ULTRASONIC SENSORS

Trabajo de integración curricular presentado como requisito para la obtención de título como Ingeniero en Tecnologías de la Información

Autor:

Jordan Rodrigo Montenegro Cárdenas

Tutor:

Oscar Chang, PhD

Urcuquí, septiembre, 2021

Urcuquí, 2 de agosto de 2021

Digitally signed by GUSTAVO ADOLFO COLMENARES PACHECO
DN: cn=GUSTAVO ADOLFO COLMENARES PACHECO c=EC l=QUITO o=BANCO CENTRAL DEL ECUADOR ou=ENTIDAD DE CERTIFICACION DE INFORMACION-ECIBCE
Reason: I'm the author of this document
Location:
Date: 2021.08.16 14:04:05:00

SECRETARÍA GENERAL
(Vicerrectorado Académico/Cancillería)
ESCUELA DE CIENCIAS MATEMÁTICAS Y COMPUTACIONALES
CARRERA DE TECNOLOGÍAS DE LA INFORMACIÓN
ACTA DE DEFENSA No. UITEY-ITE-2021-00025-AD

A los 2 días del mes de agosto de 2021, a las 09:00 horas, de manera virtual mediante videoconferencia, y ante el Tribunal Calificador, integrado por los docentes:

Presidente Tribunal de Defensa	Dr. IZA PAREDES, CRISTHIAN RENE , Ph.D.
Miembro No Tutor	Mgs. COLMENARES PACHECO, GUSTAVO ADOLFO
Tutor	Dr. CHANG TORTOLERO, OSCAR GUILLERMO , Ph.D.

El(la) señor(ita) estudiante **MONTENEGRO CARDENAS, JORDAN RODRIGO**, con cédula de identidad No. **0402041131**, de la **ESCUELA DE CIENCIAS MATEMÁTICAS Y COMPUTACIONALES**, de la Carrera de **TECNOLOGÍAS DE LA INFORMACIÓN** , aprobada por el Consejo de Educación Superior (CES), mediante Resolución **RPC-SO-43-No.496-2014**, realiza a través de videoconferencia, la sustentación de su trabajo de titulación denominado: **Artificial inteligence based position locator using ultrasonic sensor.** , previa a la obtención del título de **INGENIERO/A EN TECNOLOGÍAS DE LA INFORMACIÓN**.

El citado trabajo de titulación, fue debidamente aprobado por el(los) docente(s):

Tutor	Dr. CHANG TORTOLERO, OSCAR GUILLERMO , Ph.D.

Y recibió las observaciones de los otros miembros del Tribunal Calificador, las mismas que han sido incorporadas por el(la) estudiante.

Previamente cumplidos los requisitos legales y reglamentarios, el trabajo de titulación fue sustentado por el(la) estudiante y examinado por los miembros del Tribunal Calificador. Escuchada la sustentación del trabajo de titulación a través de videoconferencia, que integró la exposición de el(la) estudiante sobre el contenido de la misma y las preguntas formuladas por los miembros del Tribunal, se califica la sustentación del trabajo de titulación con las siguientes calificaciones:

Tipo	Docente	Calificación
Miembro Tribunal De Defensa	Mgs. COLMENARES PACHECO, GUSTAVO ADOLFO	10,0
Tutor	Dr. CHANG TORTOLERO, OSCAR GUILLERMO , Ph.D.	10,0
Presidente Tribunal De Defensa	Dr. IZA PAREDES, CRISTHIAN RENE , Ph.D.	10,0

Lo que da un promedio de: **10 (Diez punto Cero)**, sobre 10 (diez), equivalente a: **APROBADO**

Para constancia de lo actuado, firman los miembros del Tribunal Calificador, el/la estudiante y el/la secretario ad-hoc.

JORDAN RODRIGO MONTENEGRO CARDENAS
Firmado digitalmente por JORDAN RODRIGO MONTENEGRO CARDENAS
Fecha: 2021.08.16 16:23:38 -05'00'

MONTENEGRO CARDENAS, JORDAN RODRIGO
Estudiante

Firmado electrónicamente por:
CRISTHIAN RENE IZA PAREDES

Dr. IZA PAREDES, CRISTHIAN RENE , Ph.D.
Presidente Tribunal de Defensa

Firmado electrónicamente por:
OSCAR GUILLERMO CHANG TORTOLERO

Dr. CHANG TORTOLERO, OSCAR GUILLERMO , Ph.D.
Tutor

Mgs. COLMENARES PACHECO, GUSTAVO ADOLFO
Miembro No Tutor

GUSTAVO ADOLFO COLMENARES PACHECO

Digitally signed by GUSTAVO ADOLFO COLMENARES PACHECO
DN: cn=GUSTAVO ADOLFO COLMENARES PACHECO c=EC l=QUITO o=BANCO CENTRAL DEL ECUADOR ou=ENTIDAD DE CERTIFICACION DE INFORMACION-ECIBCE
Reason: I am the author of this document
Location:
Date: 2021-08-16 14:05-05:00

MEDINA BRITO, DAYSY MARGARITA
Secretario Ad-hoc

DAYSY MARGARITA MEDINA BRITO

Firmado digitalmente por DAYSY MARGARITA MEDINA BRITO
Fecha: 2021.08.03 16:18:29 -05'00'

Autoría

Yo, **Jordan Rodrigo Montenegro Cárdenas**, con cédula de identidad **0402041131**, declaro que las ideas, juicios, valoraciones, interpretaciones, consultas bibliográficas, definiciones y conceptualizaciones expuestas en el presente trabajo; así cómo, los procedimientos y herramientas utilizadas en la investigación, son de absoluta responsabilidad de el autor del trabajo de integración curricular. Así mismo, me acojo a los reglamentos internos de la Universidad de Investigación de Tecnología Experimental Yachay.

Urcuquí, septiembre del 2021.

JORDAN RODRIGO MONTENEGRO CARDENAS

Firmado digitalmente por JORDAN RODRIGO MONTENEGRO CARDENAS
Fecha: 2021.09.22 15:03:19 -05'00'

Jordan Montenegro
CI: 0402041131

Autorización de publicación

Yo, **Jordan Rodrigo Montenegro Cárdenas**, con cédula de identidad **0402041131**, cedo a la Universidad de Tecnología Experimental Yachay, los derechos de publicación de la presente obra, sin que deba haber un reconocimiento económico por este concepto. Declaro además que el texto del presente trabajo de titulación no podrá ser cedido a ninguna empresa editorial para su publicación u otros fines, sin contar previamente con la autorización escrita de la Universidad.

Asimismo, autorizo a la Universidad que realice la digitalización y publicación de este trabajo de integración curricular en el repositorio virtual, de conformidad a lo dispuesto en el Art. 144 de la Ley Orgánica de Educación Superior.

Urcuquí, septiembre del 2021.

JORDAN RODRIGO MONTENEGRO CARDENAS

Firmado digitalmente por JORDAN RODRIGO MONTENEGRO CARDENAS
Fecha: 2021.09.22 15:03:59 -05'00'

Jordan Montenegro
CI: 0402041131

Dedication

"To my parents: Rodrigo and Carmen
They made me what I am"

Jordan Montenegro Cárdenas

Acknowledgments

To Oscar Chang, who have been my professor and guide in this work. His support and knowledge have been invaluables in my academic formation.

To my family, without their support this job would not have been possible. They gave me the motivation to continue when the path was hard.

To my friends and housemates, who were my second family during my university stage and with who I shared unforgettable moments.

Jordan Montenegro Cárdenas

Resumen

Este trabajo consiste en la implementación de un localizador de posición basado en inteligencia artificial que utiliza Sensores Ultrasónicos y está integrado en la estructura de un pequeño robot móvil. El sistema incluye una computadora Arduino y un agente neuronal artificial auto motivado capaz de aprender por sí mismo cómo mover el robot de manera segura, en un ambiente del mundo real con obstáculos dispersos. La plataforma robótica consta de baterías, sensores ultrasónicos, motores, ruedas, etc. y está sujeta al ruido mecánico causado por la inercia de los motores, la fricción de desplazamiento de las ruedas y la imprecisión de los sensores. Un agente de aprendizaje por refuerzo impulsa al robot en su fase de aprendizaje y extrae conocimientos sobre cómo moverse en una situación del mundo real. Durante la fase de aprendizaje, el conocimiento se almacena en una red neuronal local entrenable que se ejecuta en tiempo real en la placa de la computadora Arduino local y aprende la política requerida para mover el robot. Esta red está equilibrada y entrenada para satisfacer la condición de avance requerida por un robot de movimiento libre. Durante la fase de operación, el robot utiliza la política aprendida almacenada en la red neuronal y se comporta como un proceso de decisión de Markov.

Palabras Clave: Robot, Auto aprendizaje, Evasión de obstáculos.

Abstract

This job consists in the implementation of an Artificial Intelligence Based Position Locator that use Ultrasonic Sensors and is integrated in the structure of a small mobile robotic car. The system includes an Arduino computer and a self-motivated artificial neural agent capable to learn by itself how to move the robot safely around, in a real world platform with scattered obstacles. The robotic platform comprises batteries, ultrasonic sensors, motors, wheels etc. and is subjected to mechanical ambient noise cause by the motors inertia, wheels' displacement friction and sensors imprecision. A reinforcement learning (RL) agent drives the robot in its learning phase and extract knowledge about how to move in a real world situation. During the learning phase, knowledge is stored in a trainable local neural network that runs in real time in the local Arduino computer board and learns the policy required to move the robot. This net is balanced trained as to satisfy the move ahead condition required by the a free moving robot. During the operation phase, the robot uses the learned policy stored in the neural net and behaves as a Markov decision process.

Keywords: Robot, Self-taught, Obstacle avoidance.

Contents

List of Tables

List of Figures

Chapter 1

Introduction

1.1 Background

Artificial intelligence have been of interest of the researches along the world in the recent decades. Artificial intelligence arose as research discipline in the Dartmouth Summer Research Project on Artificial Intelligence [1] in 1956. The concept of intelligence machines was strongly influenced by Alan Turing on its paper "Computing Machinery and Intelligence" published in 1950 [2]. In this paper, Turing propose the idea of machines that are able to learn and that can equalize or overcome human performing in intellectual tasks such as chess game.

From this point and thanks to the development of other hardware improvements such as the transistor that allowed the computer processing improvements, there have been enormous advances, and IA has found application in all the fields. Researches in medicine, sports, business, education, and other fields are the fields in which IA has been applied, and in all of them, it has acquired exceptional results in a short time. A special section of IA is reinforcement learning (RL), an approach based on the conductivism psychology theory. This theory was proposed by J. Watson in 1913 [3] and proposes that the environment determines the behavior of entities such as humans and animals. Reinforcement learning studies the interaction between an agent and a dynamic environment; the agents try to figure out how to behaves in the environment through trial-an-error. [4]. Since the beginning of RL, it has been applied in many fields and solved all kinds of problems. Rl has been used to solve problems in physics [5], chemistry [6, 7], protein folding [8]; among others.

Among all the fields of applications, one of huge interest s the robot development field, also known as robotics. Robotics is related to human functions' emulation by using mechanisms, sensors, actuators, and computers [9]. A robot might be defined as a machine that can be controlled either by an external source or by an inter source. In the early stages of robotics, the operations and control of a robot were done entirely with explicit programming, i.e., the robots had stored in its memory how to behave precisely at any time. The storage needed to save all this information was acceptable when the task was not too complex, or the size of the environment was small or medium. However, across the time, it was visibly the need for more complex tasks. That is why the advance of artificial

intelligence and robotics integrated into one field to produce robots that can learn human behavior and imitate it. Some of the function imitated by robots includes clever playing games such as clever Tic Tac Toe gaming [10, 11] or self-driven vehicles [12]. This last field is of vital importance since it have evolver the transporation technology.

1.2 Problem statement

Although the development of the artificial intelligence algorithms have been huge in the recent years, there are still some issues that needs to be solved. In the case of the self-driving robotic cars some of those issues relies in the way that artificial intelligence is integrated with them.

One concern is related with the computational power needed to execute the artificial intelligence algorithms. Nowadays, there are different IA algorithms that may be used to implements self-driving robotic cars, for instance neural networks, fuzzy logic, Q-learning and so on. However, due to the complexity of the tasks, many of these algorithms might require a lot of time and processing power to get an effective output.

Usually, the requirements of significant processing power are not a problem when working in average-sized workstations such as laptops or desktop computers because those devices are designed to support average workloads and are provided with storage size in the magnitude of gigabytes. For this thesis's purposes, it was required to confine all the required hardware of a self-driven car in an Arduino board, which has to be autonomous in terms of power supply and its computational power, besides acquiring knowledge through the exploration of its environment. To summarize, robotics cars need to have a very efficient computational arrangement that provides enough storage and supporting artificial intelligence algorithms and finally works without external influence other than its environments. The current degree project aims to fulfill all these requirements. On the other hand is very important from the AI point of view to develop systems and/or robots that learn by themselves in an unsupervised way, that its the machine has to figure thing out by itself. This study develops a robotic environment where a reinforcement learning agent explores a physical ambient and learns to move around without human supervision

1.3 Objectives

1.3.1 General Objective

Build a mobile platform equipped with ultrasonic sensors driven by artificial intelligence algorithms which can accomplish the First level of the Brooks layered system.

1.3.2 Specific Objectives

- Build a mobile platform provided with ultrasonic sensors and motors that is autonomous.

- Design an artificial intelligence algorithm with the capacity of self learning and able to guide the movements of the mobile platform.
- Embed an artificial intelligence algorithm in an small and cheap micro controller such as Arduino board

Chapter 2

Theoretical Framework

In this chapter, the theoretical background of the concepts used in developing the present work is depicted. The first part of the chapter is dedicated to the hardware elements; meanwhile, the second part is used to detail the software elements.

2.1 Arduino Platform

The Arduino Platform [13] is an easy-to-use open-source platform that includes hardware and software. Many kinds of Arduino boards represent the hardware part. Each board is equipped with a microcontroller and several input/output pins to connect several transducers and sensors. There is a huge variety of sensors to scan different environment variables such as temperature, humidity, light, UV radiation, sound, distance, etc. The software part is an open-source IDE used to write C++ code to program the microcontroller in the Arduino board.

There are several types of Arduino boards; they differ by the board's size, the number of inputs and outputs, the size of flash memory, or even the size of EEPROM. The Arduino board used in the present work is described following.

2.1.1 Arduino Mega 2560

The Arduino Mega 2560 is a microcontroller which works using the ATmega2560 microcontroller, an 8-bit Atmel Microcontroller with 16/32/64KB In-System Programmable Flash. The characteristics of the Arduino Mega are detailed in Table 2.1.

The main component of the board is the microcontroller which represents the board's processing unit . The microcontroller is the part of the board that is programmed, and it contains the Flash memory, the EEPROM, and the SRAM. The Flash memory is where the program will be stored and includes a bootloader where the initial configuration of the board is set. The EEPROM (Electrically Erasable Programmable Read-Only Memory) is a memory used to store information permanently on the board. The main of componenets of the Arduino Mega 2560 microcontroller [14] is shown in Figure. 2.1.

Microcontroller	ATmega2560
Operating Voltage	5V
Input Voltage (recommended)	7-12V
Input Voltage (limit)	6-20V
Digital I/O Pins	54
Analog Input Pins	16
DC Current per I/O Pin	20 mA
DC Current for 3.3V Pin	50 mA
Flash Memory	256 KB
SRAM	8 KB
EEPROM	4 KB
Clock Speed	16 MHz
LED BUILTIN	13

Table 2.1: Arduino Mega 2560 Features

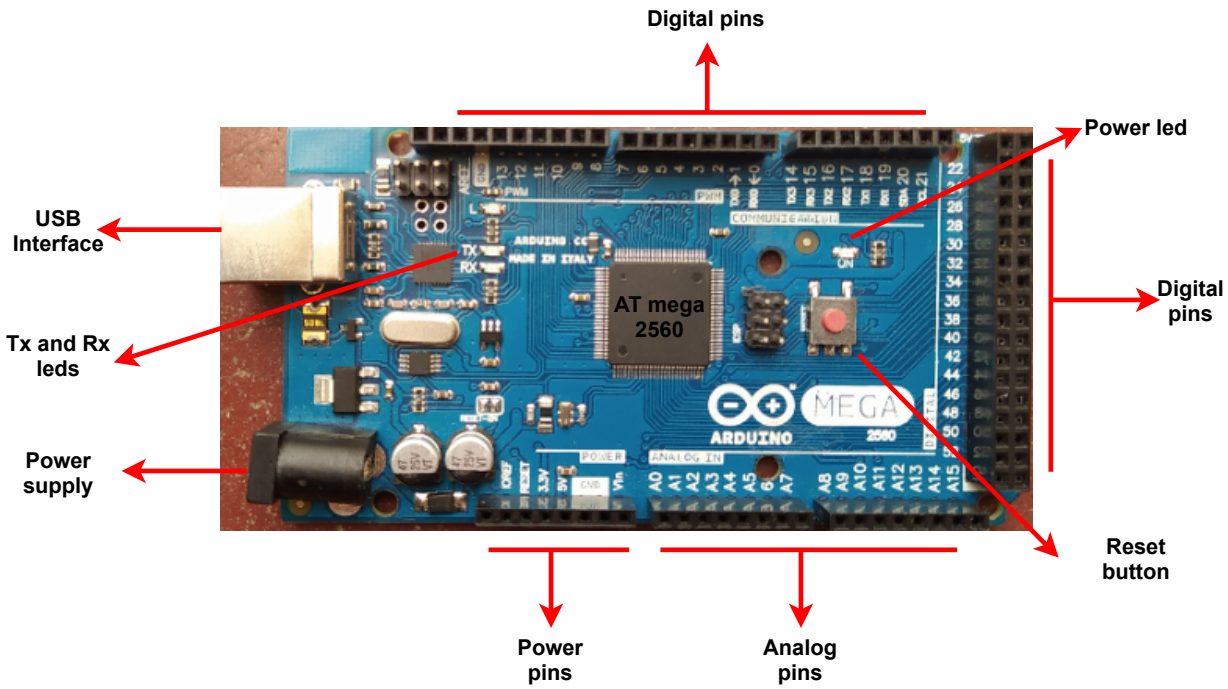


Figure 2.1: Arduino Mega 2560 main components.

2.2 Ultrasonic Ranging Module HC - SR04

It is an electronic device that is used as a proximity sensor. Two elements compose this device: the transmitter, which emits the sound wave at a frequency of 40 KHz; and the

receiver, which receives the sound wave after it has traveled and bounced from an object [15]. In this way, the sensor allows computing the distance at which an object is found by computing the time between the sending and the reception of the sound waves, as shown in Figure. 2.2. The formula used to compute the distance d in cm is shown in Eq. 2.1, where t is the time measured by the sensor.

$$d = \frac{t}{59} \tag{2.1}$$

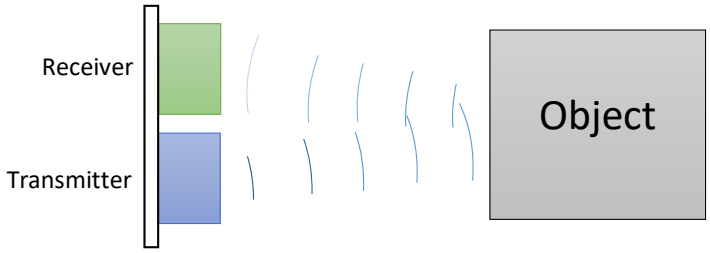


Figure 2.2: Working of ultrasonic sensor HC-SR04

Due to the sound wave's behavior, there can be some troubles when detecting objects which angle of incidence is not parallel to the sensors, as can be seen in Figure. 2.3. In this case, the incidence angle causes the reflected wave to not arrive at the receiver, leading to errors in the measures.

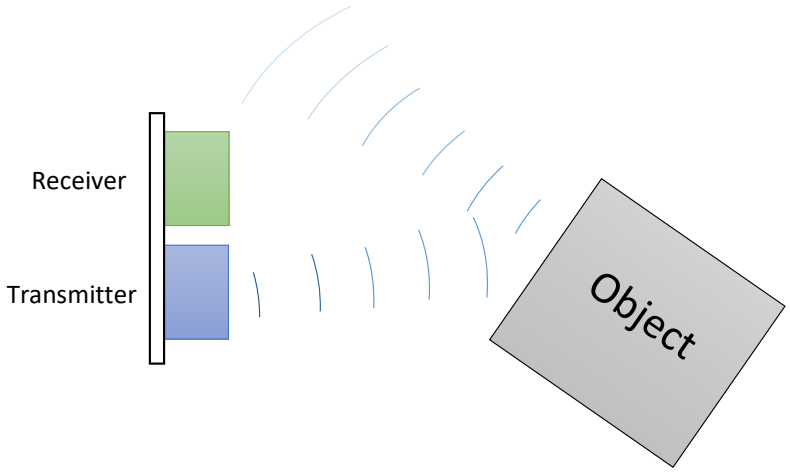


Figure 2.3: Problem with sensor in different angles

2.3 H-Bridge L293B

An H-Bridge is a circuit that allows controlling the movement of DC motor. The basic schema is illustrated in Figure. 2.4. Depending on which switch are closed, the DC motor will move forward or backward.

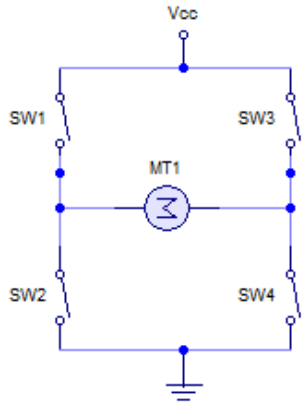


Figure 2.4: H-bridge.

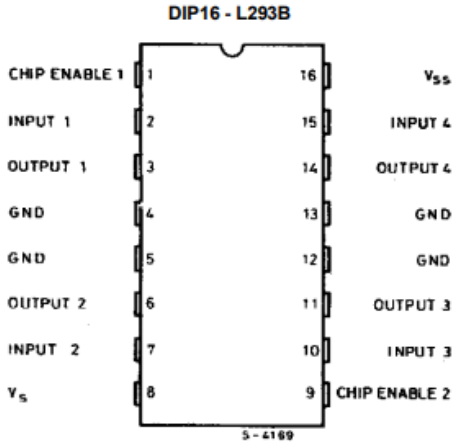


Figure 2.5: L293B Pins Distribution

The H-bridge L293B is an integrated circuit (IC) that includes two H - bridges. It has four channels; each one has an input value and an output. The distribution of this IC's pines is shown in Figure. 2.5 [16]. The backward and forward movements of a DC motor can be controlled using two channels of this IC. This device has a maximum output per channel of 1A, and this is why it is very well used in electronic and robotic projects due to its high performance.

2.4 DC Motor

A DC motor is an electronic component that transform electric energy in mechanical energy. It works with direct current (DC) [17] . A motor has two connections, one is for the ground(negative) connection and the other is for the positive terminal. Depending of the polarization of a DC motor it will move in one direction or another.

2.5 Artificial intelligence algorithms

The artificial intelligence algorithms can be classified depending on the learning approach that implements to achieve a certain object. The main categories are: supervised learning, unsupervised learning, and reinforcement learning

2.5.1 Supervised learning

This learning approach is made with an input dataset and its corresponding target outputs. The main objective is to minimize the error between the network output and the target output. This process is done in the network's training step that consists of iterative computing and adjusting the ANN weights. Once the ANN produces a satisfactory output for the input, the training ends, and the weights are fixed so the network can be put in operation in the test phase [18].

2.5.2 Unsupervised learning

In this learning paradigm, the neural network is not provided with a target output; just the input dataset is given. The networks try to discover patterns or trends in the input data without an external teacher signal. This learning is also called self-organized learning since they establish a task-independent measure to evaluate the quality of representation that the network is required to learn[18]. Some commons applications of unsupervised learning include [19] clustering, pattern configuration, principal components analysis.

2.5.3 Reinforcement learning

The main feature of reinforcement learning is based on the logical principle that if an action is followed by a satisfactory state of affairs or by an improvement in the state of affairs, then the tendency to produce that action is strengthened, i.e., reinforced [20]. In this paradigm of ANN learning, the learning machine performs a certain action on its environment and gets a feedback response from the environment. Based on this response, the learning machine grades its action as good (rewarding) or bad (punishable) and adjusts its parameters [21]. The reinforcement learning problem can be summarized in an entity's problem learning from interaction to achieve an objective or a goal. The entity that learns and that is the decision-maker is known as an agent. Meanwhile, everything else that interacts with an agent is called the environment [22].

2.6 Artificial neural networks

Artificial neural networks have been developed as generalizations of mathematical models of biological nervous systems. The basic processing elements of neural networks are called artificial neurons. The basic working of an artificial neuron can be explained with an analogy to a biological neuron: the synapse and its effects are represented by the connection weights that modulate the effect of the associated input signals, and the nonlinear behavior

exhibited by neurons is represented by a transfer function[23]. In this way, the neuron impulse is computed as the weighted sum of the input signals, being transformed by the transfer function, as can be seen in Figure.2.6. The mathematical representation of a neuron's working is detailed in Eq. 2.2.

$$O = f\left(\sum_{i=1}^{n} w_i \cdot x_i + b\right) \tag{2.2}$$

Where f is the activation function, also known as transfer function. Each x_i corresponds to an input and each w_i to its respective weight. A threshold is added to the weighted sum in order to determine the degree of sensitivity of a neuron to be inhibited or activated [24]. The learning capability of an artificial neuron is achieved by adjusting the weights in accordance to the chosen learning algorithm.

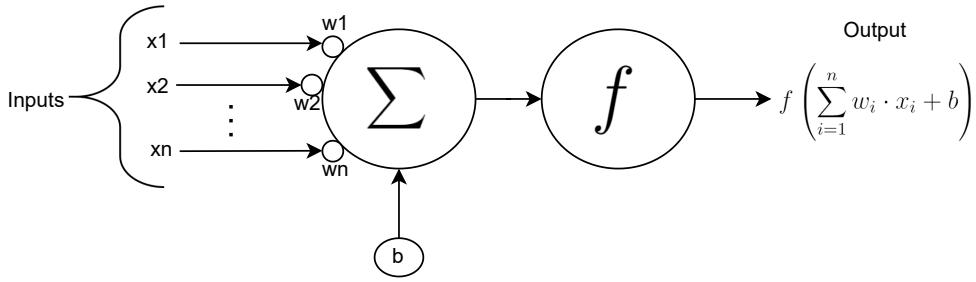


Figure 2.6: Artificial neuron functions

2.6.1 Activation Functions

Activation functions also knows as transfer functions are used in artificial neural networks to transform the weighted addition of the inputs $\sum_{i=1}^{n} w_i \cdot x_i + b$ into the output of a particular layer and supply it as input to the next layer [25]. An activation function is used to get a non-linear behavior that emulates the process of activation synapses in biological neurons. The activation function allows the neural network to identify and learn complex mappings from data. Several functions can be used as activation function; the most common are described following:

Binary Step Activation Function

This function is based on a threshold, i.e., if the function's input is greater than a certain threshold, the neuron is activated; otherwise, the neuron will be deactivated. Due to its simpleness, this function only is recommended when working with binary classification tasks rather than multiclass. Also, the derivative of this function is zero, which represents a problem in the neural network's learning algorithm. This function is shown following .

$$f(x) = \begin{cases} 1 & \text{, if } x >= 0 \\ 0 & \text{, if } x < 0 \end{cases}$$

Linear Activation Function

This function is shown in Eq. 2.3, the output is directly proportional to the input. The derivative of the function is equal to the a value; that is why the updating factor of the weights will be the same in every iteration. Due to its characteristics, this function is not useful when dealing with complex data, but it can be used for simple tasks.

$$f(x) = a \cdot x \tag{2.3}$$

Sigmoid Activation Function

This function is shown in Eq. 2.4. It is the most used activation function in neural networks. It has some benefits such as its output values are in the range [0,1].

$$f(x) = \frac{1}{1 + e^{-cx}} \tag{2.4}$$

The derivative of this function is in Eq, 2.5. Since the learning algorithm is based in the derivative of the activation function, it is an important factor. In the case of the sigmoid function, its derivative allows the learning algorithm to be successful most of the times.

$$f'(x) = c \cdot f(x) \cdot (1 - f(x)) \tag{2.5}$$

Relu Activation Function

This is the rectified linear unit function, it is a non-linear activation function. The main characteristic is that with this function not all the neurons are activated at the same time which makes this function the most efficient over all the activation functions. This function is shown following.

$$f(x) = \begin{cases} x & \text{, if } x >= 0 \\ 0 & \text{, if } x < 0 \end{cases}$$

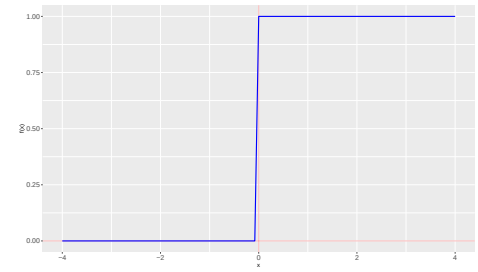
TanH Activation Function

This is the hyperbolic tangent function, it has similarity with the sigmoid as can be seen in Eq. 2.6. The difference with this function that it is symmetric with respect the origin. The range of output values is the interval $[-1, 1]$.

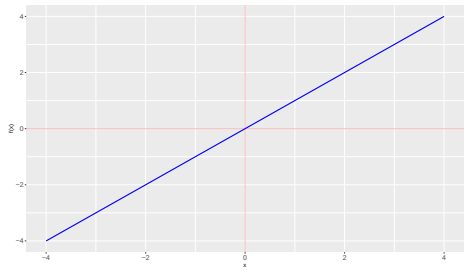
$$f(x) = \frac{e^{2x} - 1}{e^{2x} + 1} \tag{2.6}$$

2.6.2 Backpropagation algorithm

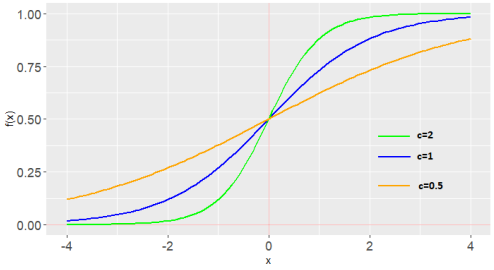
The backpropagation algorithm is the learning algorithm used by neural networks to "learn" from a training dataset. The training data set is of the form $\{(x_1, t_1), ..., (x_p, t_p)\}$ where x_i represents an input and t_i is its corresponding output. The objective of the neural network is to get the function that best approximates the x_i input with its corresponding output



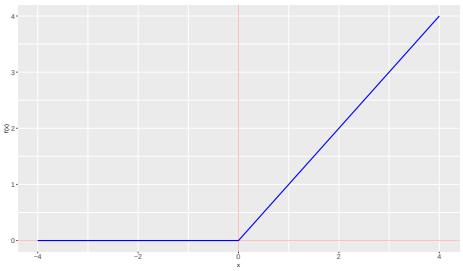
(a) Binary Step



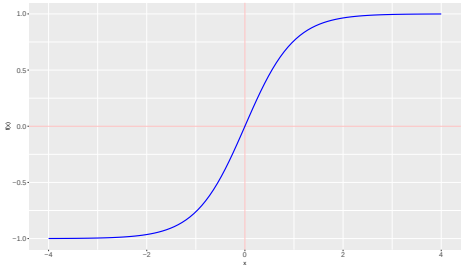
(b) Linear



(c) Sigmoid



(d) RELU



(e) TanH

Figure 2.7: Most common activation functions

t_i only by using the training dataset. As it is visible in Eq. 2.2, the output of a neural network is based in the weight w_i and the input x_i. The weights are real random numbers. When a new input is passed to the neural network, it will produce and output o_i, generally different from the t_i . The main objective of backpropagation algorithm is to minimize the cost function for all the p samples [26], generally it is the mean squared error shown in Eq. 2.7.

$$E(X, \theta) = \frac{1}{2p} \sum_{i=1}^{p} \|o_i - t_i\|^2 \tag{2.7}$$

The minimization of the error function can be done by computing the gradient of the error function $E(X, \theta)$, where x represents the p-order input vector, and θ represents the weights and the biases of the neural network. In each iteration, the gradient descent process updates the parameters θ at each iteration t using the equation shown in Eq. 2.8 [27].

$$\theta^{t+1} = \theta^t - \alpha \frac{\partial E(X, \theta)}{\partial \theta} \tag{2.8}$$

By application of the chain rule, it can be set the Eq. 2.9 where a_j^k is the weighted sum plus the bias value for nodde i in layer k, w_{ij}^k is the weight value from neurons i and j in layer k.

$$\frac{\partial E}{\partial w_{ij}^k} = \frac{\partial E}{\partial a_j^k} \cdot \frac{\partial a_j^k}{\partial w_{ij}^k} \tag{2.9}$$

We can define the backpropagated error of a neuron j as the error propagated from the last layers to the current layer k; it is shown in Eq. 2.10.

$$\delta_j^k = \frac{\partial E}{\partial a_j^k} \tag{2.10}$$

It is also possible to see that Eq. 2.11 met and can be used to the second term of Eq.2.9 .

$$\frac{\partial a_j^k}{\partial w_{ij}^k} = \frac{\partial}{\partial w_j^k} \cdot \left(\sum_{l=0}^{r_{k-1}} w_{lj}^k \cdot o_l^{k-1} \right) = o_i^{k-1} \tag{2.11}$$

Then it is possible to use Eq. 2.10 and Eq.2.11 and replace it in Eq. 2.9 resulting in Eq. 2.12 .

$$\frac{\partial E}{\partial w_{ij}^k} = \delta_j^k \cdot o_i^{k-1} \tag{2.12}$$

The backpropagation algorithm for a multilayer neural network can be divided in the following steps [24]:

i) Feed-forward computation

The input vector X is presented in the input of the network. The output o for each neuron in all the layers is computed according to the Eq. 2.2.

ii) Backpropagation to output layer

Following the Eq. 2.12, it is necessary to compute the backpropagated error δ_j^k. After the partial derivative is applied, the backpropagated error can be computed using the Eq. 2.13 where f' represents the first derivative of the activation function and a_1^m represents the weighted sum of the neuron plus the bias.

$$\delta_j^m = (t_j - o_j) \cdot f'(a_j^m) \tag{2.13}$$

iii) Backpropagation to the hidden layers

In this step, the computation of the weights of the hidden layers is computed. This process begins with the hidden layer connected to the output layer and advances towards the layer connected to the input layer. After the partial derivative is applied, the backpropagated error can be computed using the Eq. 2.14.

$$\delta_h^k = f'(a_h^m) \cdot \sum_{l=1}^{r_{m+1}} w_{lh}^{m+1} \cdot o_l^{m+1} \tag{2.14}$$

iv) Weights updates

The final step of backpropagation algorithm is to update the weights W and the biases b with the Eq. 2.15 and Eq.

$$\Delta w_{ij}^k = -\eta \cdot \frac{\partial E(X, \theta)}{\partial w_{ij}^k} \tag{2.15}$$

2.6.3 Epochs

An epoch is defined as one iteration of the training process where it is performed for all the elements in the training set $(x1, x2, ..., xp)$ [28].

2.6.4 Training Styles

There are two ways in which the training process, i.e., how parameters θ of a ANN might be updated: online and batch training. The difference in these techniques lies in the time when the parameters θ are updated.

Online training

The parameters θ are updated after one unit of the training set is passed to the neural network. In this way, the weights and the biases are updated after every cycle of feedforward and backpropagation [29]. This training is done in reinforcement learning, where the training set is acquired meanwhile an agent acts over an environment.

Batch training

In this kind of training, the parameters θ updating is done after a determined number of training set elements are passed through a feed-forward and backpropagation cycle. This number is called the batch size [30]. During this training, each training set element's gradients are summed and updated when the batch size is achieved.

2.6.5 Issues with ANN

When training an ANN there is neccesary to take into account some events that may leads in poor quality training

Under-fitting

This issue usually occurs when the ANN was not able to learn appropriately from the training data in the training phase. This event results in an imperfect ability of the ANN

to predict new inputs. The resultant classifier will have no predictive power or map the training data correctly [31].

Over-fitting

It is a very well know problem when working with ANN. Over-fitting happens when in the training phase, the network does not enhance its capacity to solve the problem anymore. However, it begins to learn specific features of the training dataset and lose its generalization capacity when new data arrives [32].

2.7 Markov Decision Process

A Markov Decision Process (MPD) aims to state the components of reinforcement learning formally: states, agents, and rewards, shown in Figure. 2.8, and to find a solution to it [33]. An MPD is used to solve the problem of choosing the sequences of decisions that will maximize the expected return obtained from a process [34]. The process is composed of a determined number of stages t that are members of a set T. The whole process is controlled by a decision-maker system that acts in each t stage and can affect the process's evolution through time. The set T might be either finite or infinite and either a discrete set or a continuum. When it is discrete and a finite set, the process is known as a Finite Markov Decision Process, and its T set is of the form $T = 1, 2, ..., N$ where N is the number of stages of the process. The agent takes the decision maker's role.

Formally, we can define an finite MPD as a set of objects of the form (T, S, A, $p_t(j|s,a)$, $r_t(s,a)$). Where S is the set of possible states of the process. A is the set of actions. $p_t(j|s,a)$ is transition probability function i.e. the probability that the system is in state j if action a is chosen in state s at time t. $r_t(s,a)$ is the expected reward received after performing a action in state s at time t [35].

There is possible to define a decision rule which is a function $d_t : S_t \rightarrow A_{s,t}$ that determines the action that the agent will take when it is on state S_t. Each decision rule is Markovian since it only depends on the current state and not of the past. The set of the possible decision rules at time t is denoted as D_t and is known as the decision set. In order to solve a finite MPD, it is necessary to define what a policy is. A policy is a chain of decision rules $\pi = d1, d2, ..., d_N$ used for the agent to make a decision at each time t. Having $v_n^\pi(s)$ as the expected total reward over the n stages of MPD where policy π is used and the system begins in state s. The final goal is to find an optimal policy $\pi*$ such that it produces the largest expected total reward [22]. $v_n^\pi(s)$ is defined in Eq. 2.16.

$$v_n^\pi(s) = E_\pi \left[G_t | S_t = s \right] = E_\pi \left[\sum_{k=0}^{n} \gamma R_{t+k+1} | S_t = s \right] \tag{2.16}$$

2.8 Layered control system for a mobile platform

The mobile platform's control is based on the layered system proposed by R. Brooks in [36]. The diagram of the control system is depicted in Figure. 2.9. The higher-level layers

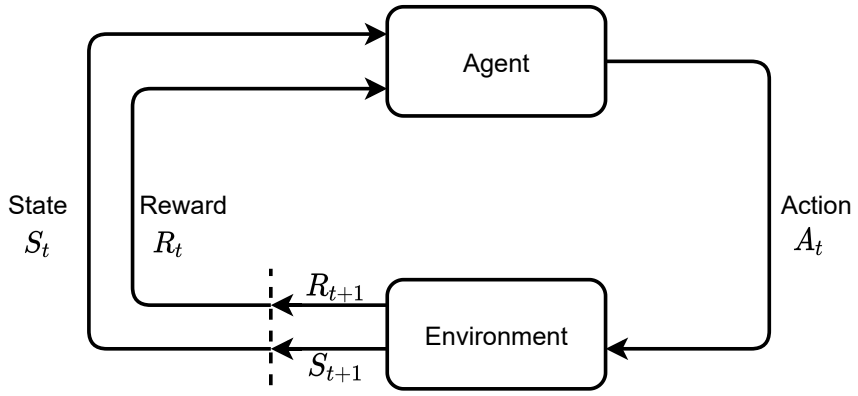


Figure 2.8: Agent - environment interaction

subsume the lower-level layers. In this way, the mobile platform can work well with the lowest layer. The addition of the superior layer will add several functionalities to the mobile platform's behavior. This layered control system was implemented for the first two layers with explicit programming.

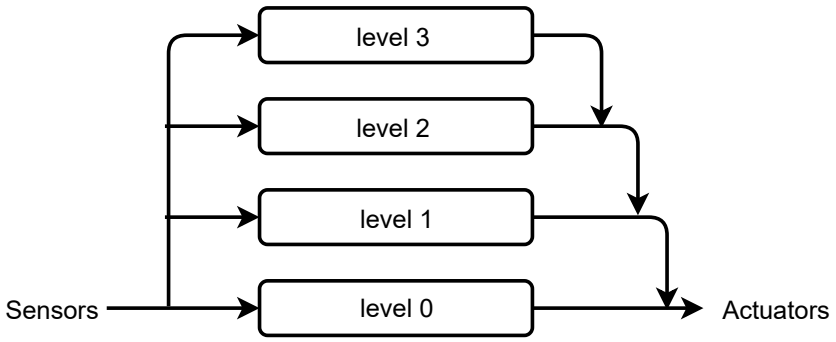


Figure 2.9: Layered control system

2.8.1 Zero level

This is the lowest layer, and its function is that the robot does not come into contact with other objects, i.e., it ensures that the robot kept to a certain distance of the nearby objects. The way the mobile platform achieves the Zeroth level moves away if something gets close to the robot or stops if an object is too close to the mobile platform. This is the first aim that the mobile platforms must accomplish over other conditions: wander around, target searching, etc.

2.8.2 First level

The aim of this layer, with the zero layer subsumption, is to provide the robot with the ability of wander around an environment environment avoiding obstacles.

Chapter 3

State of the Art

Artificial intelligence applied to avoiding obstacles in mobile platforms has been widely studied in recent years, achieving auspicious results. Many techniques have been carried out to build mobile platforms to learn how to move in a space avoiding obstacles.

3.1 Artificial intelligence applied to avoid obstacles in mobile platforms

One of the most popular algorithms in robotics is Q-learning due to its powerful features. However, one of the issues with Q-learning lies in the Q-matrix size, which directly depends on the number of states and actions. [37] deals with this problem by limiting the number of states and actions. Their results are quite good since they achieve path planning correctly with both targets and obstacles as mobile objects. However, the training process is done knowing the exact position of the obstacles and the target.

Another approach to attack the problem is by using Artificial Neural Networks (ANN). [38] implements a multi-layer perceptron with 3 inputs corresponding to the distance measured by three ultrasonic sensors and four outputs corresponding to the possible movements: left, right, forward, and backward.

In [39], the authors implement self-learning neural networks that control a mobile robot to get a target avoiding obstacles and especially concave U-shaped obstacles. This work is based on the algorithm and simulation experiments proposed by [40]. The proposal is a topologically organized neural network in which the separate behavioral equation describes each neuron. In this way, the best trajectory is generated in real-time through the dynamic activity 2D - landscape of the neural network by finding the neighboring neurons with the highest activation towards a target.

Another very widely used approach is genetic algorithms(GA). For instance, in [41] authors propose a path planning system for mobile robots based on genetic algorithms. In this model, they get a target avoiding static and mobiles obstacles. Their simulation results

show the success of GA when dealing with path planning and obstacle avoidance. In [42] the authors developed a GA based algorithm to control the navigation of a mobile robot that is able to escape from dead-end zones. This is done by modifying the chromosome structure and providing the robot with a memory that stores the sensor information, the number, orientation, and coordinates of the robot in every step. They test their algorithm in simulations achieving escaping from dead-end zones in all the simulations.

Fuzzy logic algorithms have also been applied to avoid obstacles in mobile platforms. That is the case of [43] where the author compares the performance in a static environment of fuzzy navigation against three classical approaches for obstacle avoidance (potential field method, vector field histogram plus method, and Local navigation method). The fuzzy navigation achieved the fastest results and had the easiest implementation.

Other approaches try a combination of several techniques to get better results. [44] presents a method that combines Q-learning with a neural network planner to do robot path planning in dynamic environments. They present simulation and real experiments that achieved good results when the robot gets the target in no more than a minute and 15 iterations of the algorithm. A similar combination of these algorithms is done in [45] using ultrasonic sensors as inputs and discrete movements as outputs. This work is divided into four phases. At first, a Q-table is filled with state-action pairs. Then the ANN weights are trained with values of the Q-table. Finally, the ANN itself is trained using a reinforcement learning signal from the environment to retrain the network. They achieved better simulations results when comparing against only the Q-learning algorithm.

In [46], the authors design and implement a reinforcement learning algorithm based on an associated memory (AM) that behaves as a neural network. It uses local information of a static environment to reach a target avoiding the obstacles that are in the mid-path. The MA's input corresponds to the possible positions in which obstacles can be found, and the outputs correspond to the possible movements that can be done. Then by rewards and punishment, the MA can relate the inputs with the outputs. Finally, they achieved quite good simulation results when getting a target and avoiding obstacles.

In [47] the authors propose a combination of fuzzy control with ant colony algorithm(ACO) to attack the problem of path planning and obstacle avoidance. They compare this algorithm's behavior against a pattern search algorithm, a genetic algorithm, particle swarm optimization, and traditional ACO. Authors use ultrasonic transducers as sensing devices and achieve great results in very few iterations.

In [48], the authors implement an adaptive neuro-fuzzy inference system to control several mobile robots to reach a target avoiding obstacles in a static environment. This system combines the logic rules of fuzzy systems and a neural network's learning capability. They performed extensive simulations and real experiments with several robots achieving well results in both simulations and real experiments.

Chapter 4

Methodology

In this chapter, the proposed model is detailed along with the description of its implementation and the experiments carried out to test the model.

4.1 Mobile platform

The mobile platform was implemented using one Arduino Mega 2560 board because of its extensive digital input to include all the actuators and transducers needed. Also, the AT mega 2560 microcontroller, which is the brain of the Arduino Mega 2560, provides a vast EEPROM to store the weights of the ANN. An overview of the mobile platform is depicted in Figure. 4.2. The chassis of the mobile platform is designed to include all the elements needed to make the robot completely autonomous. This condition includes the motion actuators, the power supply, and the agent software that controls the robot. The mobile platform is equipped with a 4xAA battery holder used to power the mobile platform and a buzzer used to let the user know when the different process inside the agent begins or finish. The mobile platform's main components are grouped in three blocks: i) ultrasonic sensors blocks, ii) motors control block and iii) load/save buttons block and are described following.

4.1.1 Ultrasonic sensors block

This module includes four HC - SR04 ultrasonic sensors placed on a curve-shaped surface. The curve in the front gives the robot a better perspective to detect objects in front of it, as shown in Figure. 4.1. The schematics diagram is shown in Figure. 4.4(a). The Vcc is obtained from the Arduino board. The Trigger and Echo pin in each sensor are connected to digital input and output pins in the Arduino board, respectively, allowing the distance measurement.

4.1.2 Motors control block

This module includes the L293B C.I. and the DC motors. Its schematic diagram is shown in Figure. 4.4(b). The central part of this module is the L293B CI, which controls both

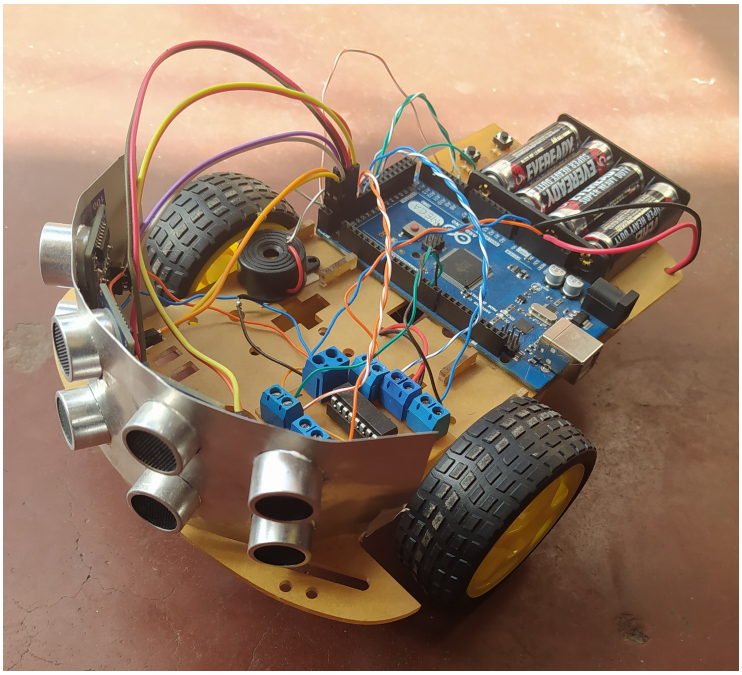


Figure 4.1: Frontal view of the robot

Input A	Input B	Function
High	Low	Clockwise
Low	High	Anti - Clockwise
Low	Low	Stop
High	High	Stop

Table 4.1: L293B H-bridge control for one Dc motor

DC motors and it is shown in Figure.4.2. PWM 1 and PWM 2 pins allow the L293B CI to control the speed of DC motor 1 and DC motor2, respectively. These pins must be wired to PWM outputs in the Arduino board. In this way, the speed of the motors can be controlled with an Arduino board. The IN1 and IN2 pins are digital input signals used to determine the directions of DC motor 1. Meanwhile, IN3 and IN4 pins are used to determine the directions of DC motor2. An H-bridge drives the rotation direction of each DC motor, the rotation's control of one DC motor is done with two digital signals, and the process is detailed in Table 4.1.

4.1.3 Load/save buttons block

This module is dedicated to communicating with the mobile platform, i.e., to give basic orders. Both buttons are installed with a pull-down resistor configuration [49], so the logical output value will be "LOW" when the button is not pressed. Button 1 is the load button that, when pushed, the robot loads the weights W vector stored in the Arduino Mega EEPROM and proceeds to begin the exploitation phase. Button 2 is called the save

button that, when is pressed, begins the routine to store the current W vector into the Arduino Mega EEPROM. This routine may be called independently, whether the robot is in the training phase or testing phase.

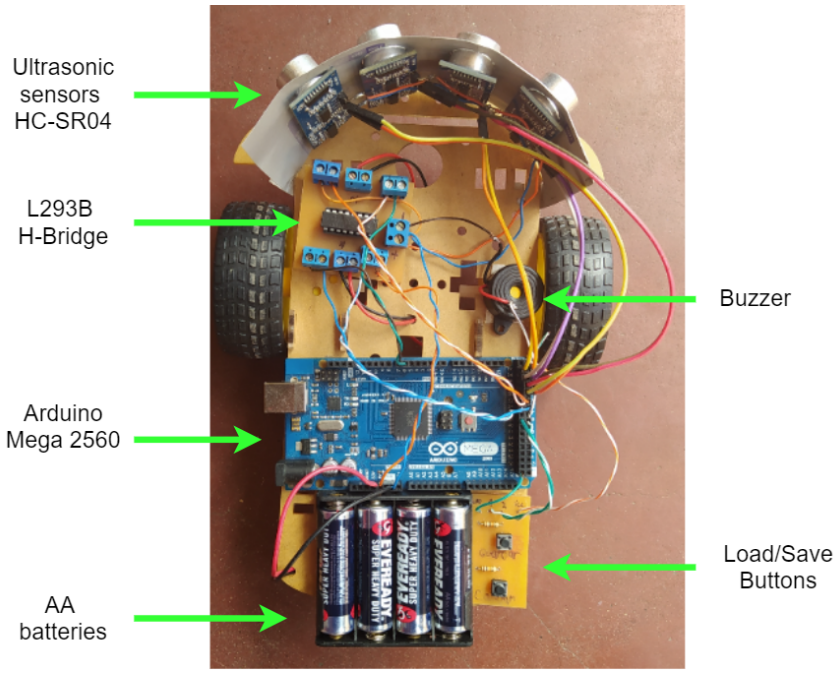
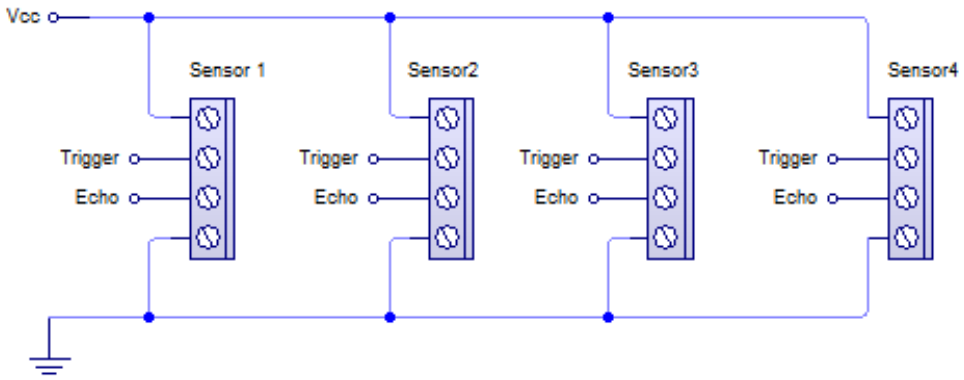
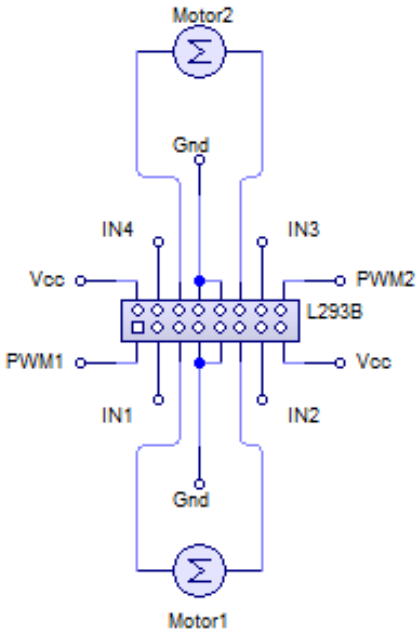


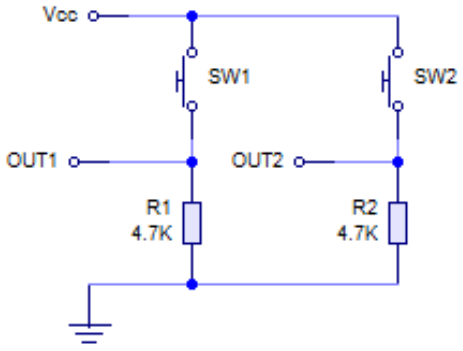
Figure 4.2: Superior view of the mobile platform



(a) Ultrasonic Sensors block

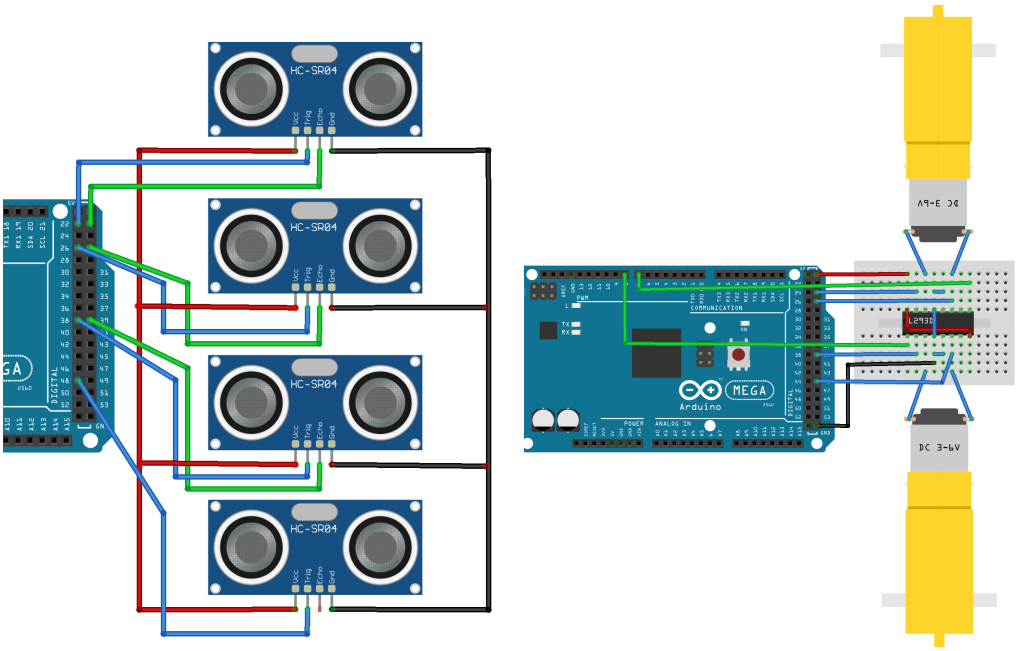


(b) L293B Block

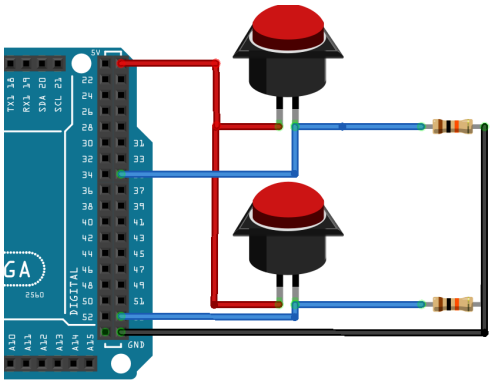


(c) Buttons Block

Figure 4.3: Schematic Diagram of the Mobile Platform Main Elements



(a) Ultrasonic Sensors block (b) L293B Block



(c) Buttons Block

Figure 4.4: Diagram of the Mobile Platform Main Elements

4.2 Software

The code developed for the entire functioning of the robot is written in C++ language. The code was developed using the Arduino IDE 1.8.13 [50], the official IDE to write C++ code and upload it into the Arduino board.

4.2.1 Neural Network Architecture

The networks architecture is shown in Figure. 4.5. The artificial neural network comprises three layers: one input layer with four neurons, one hidden layer with fifteen layers, and one output layer with six neurons. The activation functions used in all the neurons is the sigmoid function due to its advantages regarding its derivative respecting other activation functions. The c value used in the sigmoid function was set experimentally to 0.5. The learning rate η used in the backpropagation algorithm was set, also experimentally, to 0.25. The bias for all the neurons was set to 0.

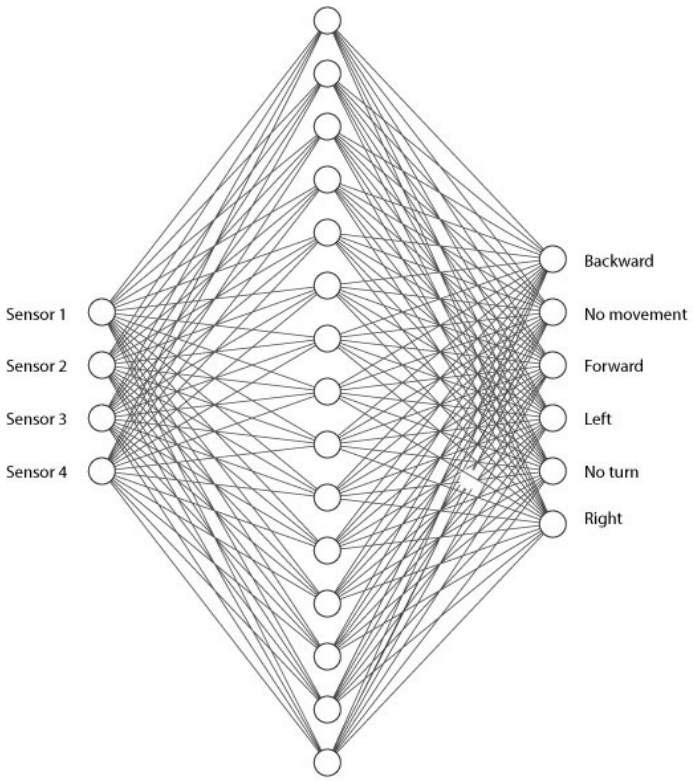
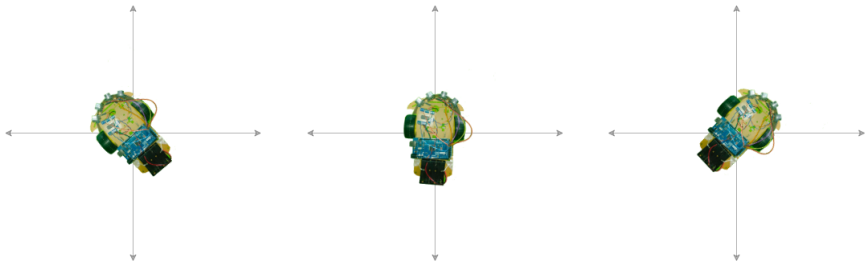


Figure 4.5: Architecture of the neural network

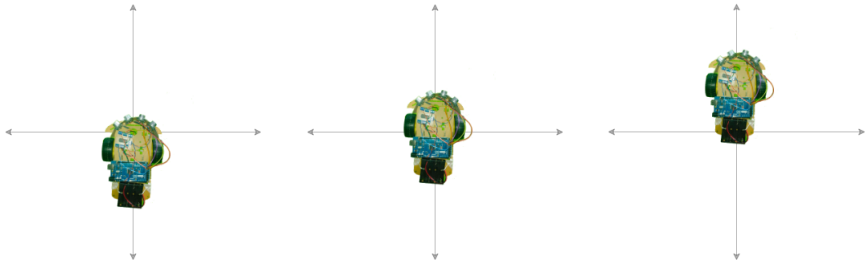
Each neuron in the input layer corresponds to one ultrasonic sensor of the mobile platform. Its value is determined depending on the achievement of the zeroth level of Brooks control system [36]. In this way, if the sensor accomplishes the zero level,i.e., there is no obstacle in its range detection, the neuron input value is set to 0; otherwise, the value is set to 1. This process is the same for all the sensors resulting in an input vector of the form (x_1, x_2, x_3, x_4). The detection range was set experimentally to $10cm$.

The neurons in the hidden layer were set experimentally to fifteen. Six neurons compose the output layer that produces the output vector of the form $(o_1, o_2, o_3, o_4, o_5, o_6)$. o_1, o_2 and o_3 represent the turn movements: left, no-turn, and right respectively and they are depicted in Figure. 4.6 . The remainder three neurons produce the output destined to control the forward movements: forward, no-movement, and backward, and they are shown in in Figure 4.7.



(a) Left (b) Quiet (c) Right

Figure 4.6: Output for turn's neurons



(a) Back (b) Quiet (c) Forward

Figure 4.7: Output for motion's neurons

4.2.2 Code structure

The code of the current degree project is divided into two main algorithms: the Algorithm 2 used in the exploration stage, a.k.a training stage, by which the robot learns, and the Algorithm 3 used in the exploitation phase, a.k.a. test phase, by which the robot uses the knowledge acquired in the exploration stage.

In addition, there is the Algorithm 1 used in the Algorithm 2. Algorithm 1 shows the so-called "Fiction Backpropagation", a backpropagation algorithm with theoretical input and target. This algorithm is in charge of teaching the robot to achieve Brooks' First level, i.e., to wander around its environment, accomplishing the zero level for all the sensors. Line 1 of the algorithm shows the target values t_i that will be used in the backpropagation algorithm. These t_i values correspond to the forward and no-turn moves of the robot. Line 2 describes the input vector passed to the ANN in the feed-forward step of the backpropagation algorithm. This vector contains the x_i values and corresponds to a Brooks' Zero level simulation in all the sensors, i.e., none of the sensors detects an obstacle nearby. Line 3 indicates that the backpropagation algorithm is performed, and it uses the input and target values set in the previous lines.

Algorithm 1: Fiction Backpropagation

1 Set forward and no-turn movement as ANN target i.e. ANN target=$(0, 1, 0, 0, 0, 1)$;
2 Set zero level as ANN input i.e. ANN input = $(0, 0, 0, 0)$;
3 Do backpropagation;

Algorithm 2 describes the training stage of the robot. At this stage, the robot performs the exploration of its environment. Line 1 shows the ANN initialization. In this step, all the weights are set with random float values between -0.5 and 0.49 since these limits achieved satisfactory experimental results. Line 2 indicates the algorithm will be executed for a determined number of max iterations specified by the user or until the load button is pressed (Algorithm 3 is invoked). Line 3 begins the robot's operation with the forwarding movement of the robot and reading of the sensors. After this movement, if the robot achieves the zero layer, nothing is interesting to learn; thus, the robot continues exploring until the zero level is not achieved. When the robot does not achieve the zero level, the process to be performed is described from Line 4 to Line 17. At first, the number of iterations is incremented; thus, the robot only counts those iterations that were useful and in which it acquired knowledge. The if-condition shown in Line 6 tells whether the two left sensors detected an obstacle. In that case, the target of the ANN is set to the vector $(0, 0, 1, 1, 0, 0)$, which indicates a right turning and a backward move. The other case, where the right sensors detect an obstacle, is shown in Line 9. In this case, the target of the ANN is set as the vector $(1, 0, 0, 1, 0, 0)$ that indicates a backward movement and a left turning. The if-conditions presented in Lines 6 and 9 are exclusive between them; in this way, the measures from the two left sensors will be read at first, and if no obstacle was detected, it will scan the measures from the two right sensors.

Once the target of the ANN was set, it is time to set the input. This process is done in Line 12, where the input vector is the one acquired in Line 3. Following, having the target and input defined is possible to perform the backpropagation algorithm shown in Line 13. At this line of the algorithm, the robot can learn the zero layer of the Brooks system, i.e., it has learned to avoid obstacles. However, the purpose of this degree project is to equip the robot with the first level of the Brooks system, i.e., with the capacity to wander around an environment; for that reason, the following line is added. Line 14 calls the "fiction backpropagation" algorithm, shown in Algorithm 1. This algorithm is used to teach the robot the first level of the Brooks layered system. Since each iteration of the backpropagation algorithm is equivalent to one epoch, two epochs are performed in each iteration of Algorithm 2. Finally, Lines 15 and 16 are used to locate the robot in a different position to continue with the exploration.

It is important to emphasize that detecting the obstacles and, therefore, the robot's capacity to stay in the zero level and avoid obstacles depends on the ultrasonic sensors' contact surface, as was mentioned previously.

Algorithm 2: Exploration or training stage

```
1  ANN Random Weights Initialization;
2  while notTrained or iterations < max_iterations do
3      Advance and read sensors;
4      if zeroLevelUnachieved then
5          iterations++;
6          if obstacleAtLeft then
7              Set backward and right movement as ANN target i.e. ANN
                target=(0, 0, 1, 1, 0, 0);
8          end
9          else if obstacleAtRight then
10             Set backward and left movement as ANN target i.e. ANN
                target=(1, 0, 0, 1, 0, 0);
11         end
12         Set current sensor values as ANN input;
13         Do backpropagation;
14         Do fiction backpropagation;
15         Step back the robot;
16         Random turn;
17     end
18 end
```

Algorithm 3 shows the exploitation phase where the robot uses the knowledge acquired in the training stage in order to wander around an environment avoiding obstacles, in other words: accomplishing the Brooks First level. Line 1 indicates that the robot will perform its process until an external factor stops it. The process begins with Line 2, where the reading of the measures of the sensors is done, resulting in the input vector. Then, the feed-forward process is performed with that input vector. Finally, the feed-forward step results in an output vector that guides the robot's movement at each iteration of the testing phase.

The output of the neural network is a vector of the form $(o_1, o_2, o_3, o_4, o_5, o_6)$ where the outputs o_1, o_2 and o_3 correspond to the neurons that control the turn movement of the robot; meanwhile the outputs o_4, o_5 and o_6 correspond to the neurons that control the backward and forward movements. The way that a winner neuron is chosen is by selecting the one with the highest value. That is, having an output vector $(0.16, 0.23, 0.35, 0.30, 0.17, 0.45)$ the winner neurons are those with the highest values for each kind of movement: 0.35 in the case of turn movements which indicated that the robot would perform a right turn and 0.45 in the case of the back and forward movement indicates the robot to move forward. The way that these movements are performed is at first the turning movement and then the forward movement. The backward movement in the exploitation phase was tuned to be less than the forward movement to improve the robot's ability to pass through narrow places.

Algorithm 3: Exploitation or testing stage

```
1 while testPhase do
2 |   Read information from sensors;
3 |   Perform feed forward with input from sensors;
4 |   Get movement output from ANN;
5 |   Execute the movement;
6 end
```

4.3 Experimental Setup

The general flowchart of the robot is shown in Figure. 4.8.

4.3.1 Environment setup

In order to evaluate the mobile platform, an environment has been built to perform both the exploration and the exploitation stages. The environment consists of a circle-shaped delimited environment that contains static obstacles with different shapes. The surface floor is made of concrete with very few irregularities, such as dust and tiny holds simulating a real scenario. The height of the obstacles and the walls is lower than $50cm$, which is more than enough for the robot to detect the objects. Since the ultrasonic sensors measure might fail depending on the surface's angle, square-shaped and circle-shaped obstacles are used to test how the mobile platform behaves with different shapes. This is done considering that the square-shaped obstacles' flat surfaces might lead to failure in the sensors' measures; meanwhile, the circle curved-shaped, and the walls are the ideal surfaces for the ultrasonic sensors.

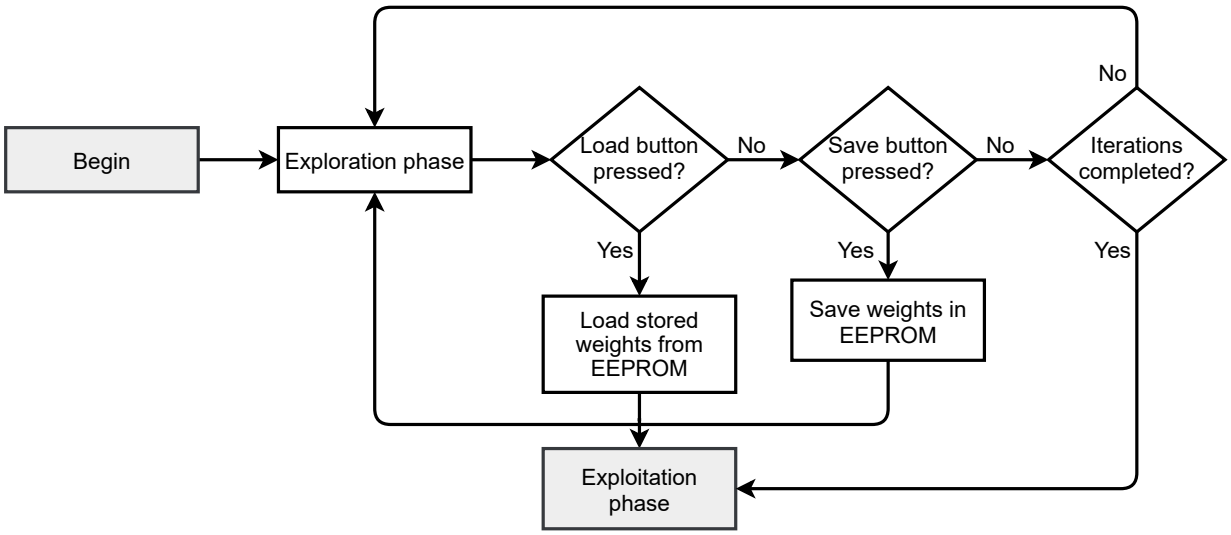


Figure 4.8: Robot's flow diagram

4.3.2 Exploration phase

At first, the robot is located at a random position in the environment, initiated in its exploration phase. Following the flow diagram shown in Figure. 4.8, at each iteration of Algorithm 2, the robot will check if any of the buttons of load or save have been pressed. Otherwise, the robot will continue to check whether the algorithm iterations have been completed or not. For training purposes, the exploration phase was done with the USB port of the Arduino Mega connected to a laptop to supervise the training and check by a human supervisor the point at which is enough trained, so the robot will be able to behave correctly in the exploitation phase. Also, this setup was set to acquire information about how the error decrease along the epochs. However, this process is just for supervision purposes since the agent algorithm allows the robot to begin the exploitation stage after a determined number of iterations have been completed.

4.3.3 Exploitation phase

The training stage, for experimental purposes, was done with the supervision of a human operator; then, once the exploration phase has acquired satisfactory results, the human operator will press the ”save weights button” to store the ANN weights in the EEPROM memory of the Arduino Mega 2560. Then, the robot's power supply is changed to the four AA batteries, and the wire connected to the laptop is unplugged since it is not necessary anymore. The exploitation phase will begin when either a user haven pressed the ”load button” or when the robot has completed with the number of max iterations previously defined.

Chapter 5

Results and Discussion

In this chapter, the results obtained from the evaluation of the mobile platform along with the artificial intelligence algorithm are shown and analyzed.

5.1 Exploration phase

The configuration used for this phase is shown in Figure. 5.1. The diameter of the environment was $1.20m$. Despite this phase can be done independently in the robot, it was plugged into the computer for human supervision. The control is done through the Serial Monitor provided by the Arduino IDE. There are two possible situations at any iteration of the robot exploration phase.

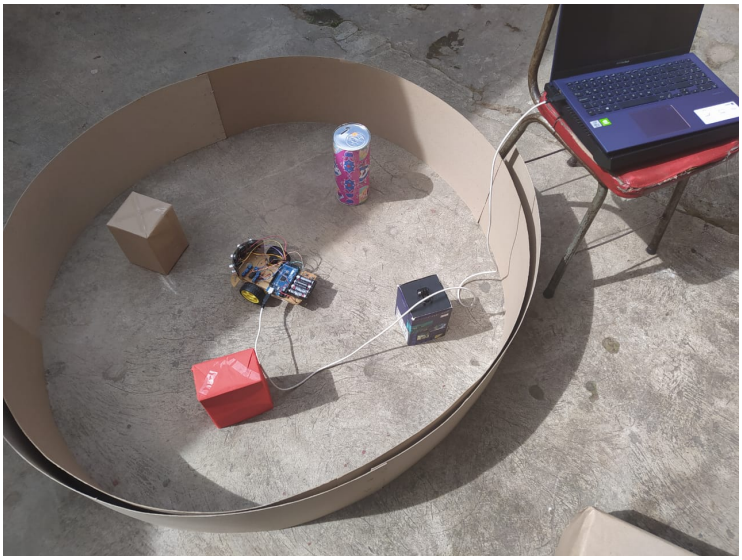


Figure 5.1: Configuration for exploration phase

In one case, the Zero layer has been achieved, so the robot keeps its exploration producing the output shown in Figure. 5.2. The first line of the output tells the current iteration of the algorithm. The second line describes the previous movement performed, and the

third line introduces the measures corresponding to the four sensors starting with the left sensor to the right sensor. In this case, since the zero layer has been achieved, the robot continues with forwarding movement until it gets out of the zero level. It is important to emphasize that the number of iterations of the training algorithm does not includes this case since it does not contribute to the teaching of the ANN.

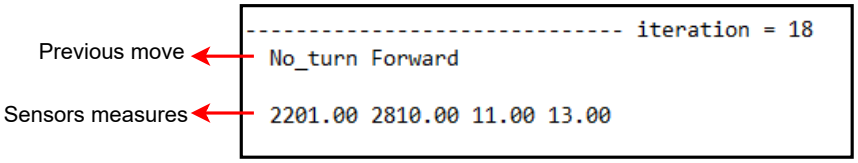


Figure 5.2: Zero level achieved

If the robot gets out of the zero level, it produces the other situation in the exploration phase, resulting in the output shown in Figure. 5.3. The first two lines indicate the same as the previous case. The third line is useful to tell the user the direction of a nearby obstacle. The fourth line confirms that the zero layer has not been achieved, and the backpropagation algorithm must be performed. The "ANN input" line describes the input presented to the ANN. The "ANN target" line describes the target presented to the ANN. The "ANN output" line shows the output produced by the neural network. The "Fiction backpro" line indicates that the fiction backpropagation has been done. After this line, the ANN target and output used in the fiction backpropagation are shown. Finally, the "Random future move" describes the future movement to be performed before the algorithm begins again.

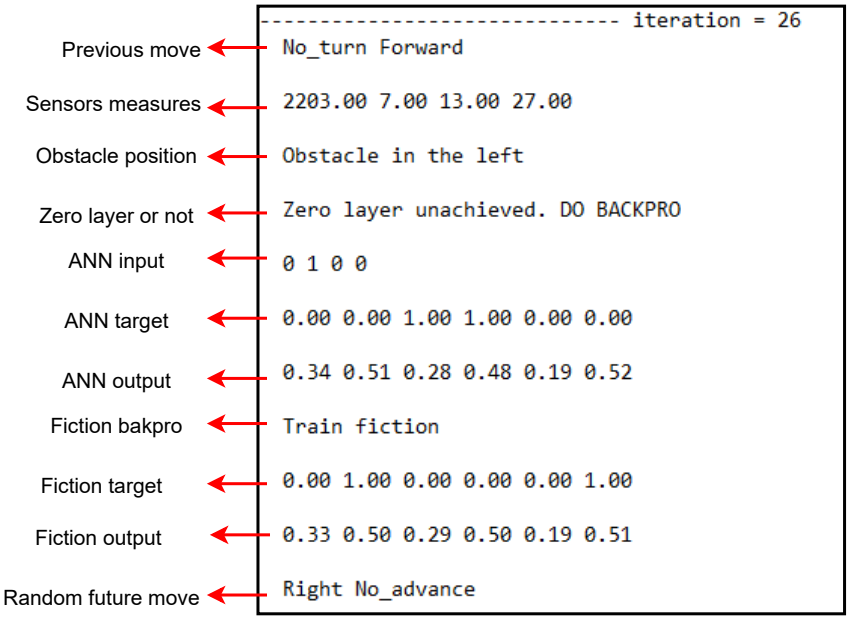


Figure 5.3: Zero level unachieved

The number of iterations needed to train the artificial neural network was 485. Taking into account that for each iteration of the training algorithm two backpropagation steps

were performed, the number of epochs was 970. At this epoch, it was possible to see that the output winner neuron agrees with the neural network's target in all the input patterns. Depending on the distribution of the obstacles and the environment's size, the training process might last more or less. If an environment is small and has many obstacles, the training algorithm will be performed more times in less time; on the other hand, if the environment is big and has few obstacles, the training algorithm will take more time. With the current environment configuration, the robot's time to perform the 485 iterations was approximately 15 minutes.

The way to analyze the algorithm's performance during the training stage is by checking the mean squared error (MSE). It is applied to study the training performance since it is the loss function used in the backpropagation algorithm in the exploration phase. MSE indicates the overall performance of the ANN at each training epoch. MSE can be seen in Figure 5.4, where the its value at epoch 0 is approximately 0.27, and across the training, it decreases to values under 0.15. The variation in the MSE during the epochs is determined by the learning rate used in the backpropagation algorithm. Choosing a different learning rate has changed the behavior of the MSE in the exploration phase resulting in worst or better results. In our case, it was proved experimentally that the learning rate value of 0.25 achieved the proper training in fewer epochs.

Since the algorithm was stopped when underfitting was avoided , i.e. when the ANN was able to correctly predict the target value, rather than when the error was minimum, the MSE does not achieve a value of 0 which means that the difference between the target and the resulting output of the ANN is extremely low.

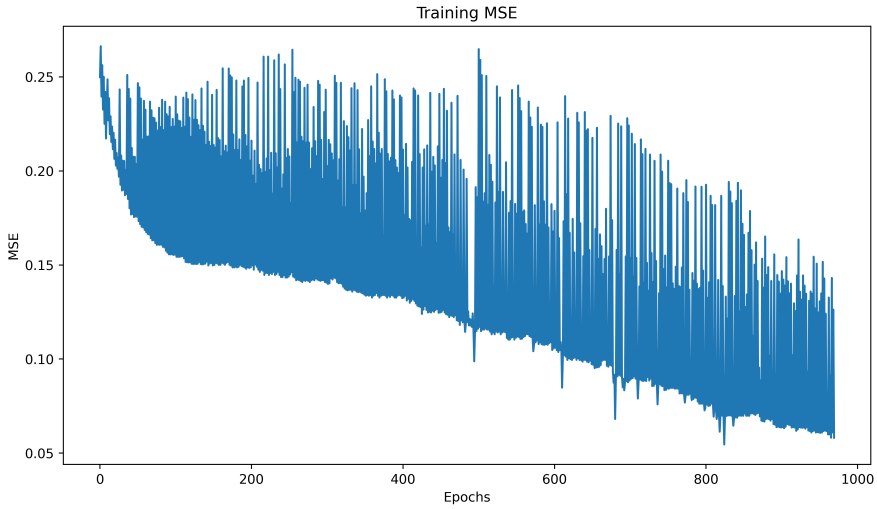


Figure 5.4: Mean squared error of exploration phase

5.2 Exploitation phase

This phase was executed in a different environment from the exploration stage; it is more extensive and includes more obstacles so the mobile platform can face several situations

different from those used in training.

The way to evaluate the algorithm's performance in this stage is by watching the behavior of the mobile platform, unplugged from the computer, in a static environment. The mobile platform is located in the environment and is triggered in its exploitation stage during the time. Two starting points were evaluated in this stage, resulting in two routes.

The first route is shown in Figure 5.5. The starting point is the position at which the robot is shown in the graph. The arrows indicate the route followed by the robot until a human operator stops it. This route begins with the mobile platform in direction to a wall. As it can be seen, this route is collision-free despite going through tight spots between an squared-shaped obstacle and the wall. After, avoiding another square-shaped obstacle it ends in the same point that it started with different direction.

The second route is shown in Figure. 5.6. This route is larger and more complex than the previous one. The repeated arrows in small spaces are caused by the robot's constant backward and forward movements until it can achieve a position in which its forward movement will not collide with any surfaces. This route presents a collision where the red x symbol is drawn. This collision was produced after the mobile platform passed successfully through a tiny space between a circle-shaped obstacle an the wall. After this, the mobile platform ended too close to the wall and due to the range of measurement of the left sensor, it failed in detecting the wall. The remainder of the route happened without collisions, and the robot avoided three squared-shaped obstacles.

Figure 5.5: First route of exploitation phase

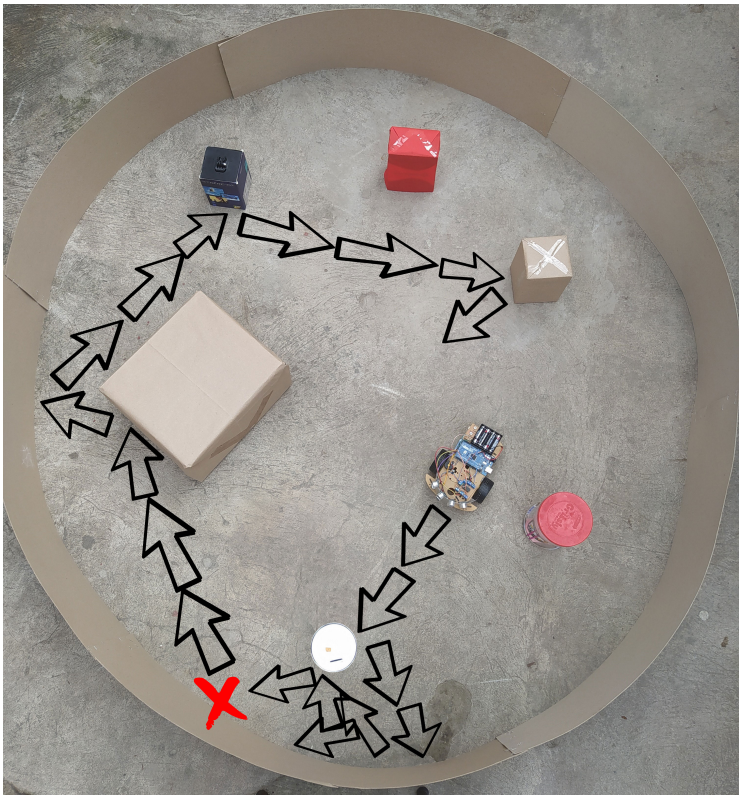


Figure 5.6: Second route of exploitation phase

Chapter 6

Conclusions

Artificial intelligence is a promising set technique capable of being applied in infinity fields, including medicine, leisure, sports, and finance. One of the most benefited fields is robotics. The innovation in artificial intelligence algorithms includes the reinforcement learning algorithms widely used in the robotic field due to their robust features, especially when dealing with self-driving vehicles. The main conclusions of the present degree project are summarized following:

1. An agent-driven approach has been implemented to control a mobile platform that can achieve the First Level of the Brooks layered control system. The main component of this approach is an artificial neural network that, after 970 epochs of online training, can guide the mobile platform movements, getting as a final result a robot able to wander around an environment successfully, avoiding obstacles.

2. The mobile platform has been built to be independent, without the need for an external factor that controls it. The robot's operation is performed in two steps. First, the self-training stage, where the robot is placed in an environment, and it can learn from continuously interacting with the environment. This phase is followed by the operating phase, where the robot applies all the knowledge acquired in the previous phase to accomplish an avoiding-obstacles behavior..

3. A 3-layer artificial neural network has been included inside an Arduino Mega 2560 board. This small, affordable, and simple board met all the requirements for implementing the mobile platform, including the necessary power supply, the input/outputs pins, the EEPROM, and the computation power.

4. The mobile platform behaves following the Markov property that includes the randomness of its whole process, including the condition that the future steps only depend on the current state.

6.1 Future Work

Several improvements will be applied to both the mobile platform and to the agent that drives it. In the mobile platform, the wheels play a central role in the mobile platform's

motion then it is necessary to equip it with wheels that have better friction with the ground surface and consequently a better motion. In terms of the software, Brooks' layered system's upper layers will be implemented to endow the robot with more complex tasks, such as get a target inside an environment.

Bibliography

[1] S. Dick, "Artificial intelligence," *Harvard Data Science Review*, vol. 1, no. 1, 7 2019, https://hdsr.mitpress.mit.edu/pub/0aytgrau. [Online]. Available: https://hdsr.mitpress.mit.edu/pub/0aytgrau

[2] A. M. Turing, "Computing Machinery and Intelligence," *Mind*, vol. LIX, no. 236, pp. 433–460, 10 1950. [Online]. Available: https://doi.org/10.1093/mind/LIX.236.433

[3] J. B. Watson, "Psychology as the behaviorist views it." *Psychological review*, vol. 20, no. 2, pp. 158–177, 1913.

[4] L. P. Kaelbling, M. L. Littman, and A. W. Moore, "Reinforcement learning: A survey," *Journal of artificial intelligence research*, vol. 4, pp. 237–285, 1996.

[5] D. Dong, C. Chen, T. Tarn, A. Pechen, and H. Rabitz, "Incoherent control of quantum systems with wavefunction-controllable subspaces via quantum reinforcement learning," *IEEE Transactions on Systems, Man, and Cybernetics, Part B (Cybernetics)*, vol. 38, no. 4, pp. 957–962, 2008.

[6] Z. Zhou, X. Li, and R. N. Zare, "Optimizing chemical reactions with deep reinforcement learning," *ACS central science*, vol. 3, no. 12, pp. 1337–1344, 2017.

[7] M. Popova, O. Isayev, and A. Tropsha, "Deep reinforcement learning for de novo drug design," *Science Advances*, vol. 4, no. 7, 2018.

[8] O. Chang, F. A. Gonzales-Zubiate, L. Zhinin-Vera, R. Valencia-Ramos, I. Pineda, and A. Diaz-Barrios, "A protein folding robot driven by a self-taught agent," *Biosystems*, vol. 201, p. 104315, 2021. [Online]. Available: https://www.sciencedirect.com/science/article/pii/S0303264720301891

[9] J. Craig, *Introduction to robotics*. Pearson Education, 2005.

[10] O. Chang, L. Zhinin-Vera, and F. Quinga-Socasi, "Self-taught neural agents in clever game playing," in *Proceedings of the Future Technologies Conference (FTC) 2020, Volume 1*, K. Arai, S. Kapoor, and R. Bhatia, Eds. Cham: Springer International Publishing, 2021, pp. 512–524.

[11] O. Chang and L. Zhinin-Vera, "A wise up visual robot driven by a self-taught neural agent," in *Proceedings of the Future Technologies Conference (FTC) 2020, Volume 1*, K. Arai, S. Kapoor, and R. Bhatia, Eds. Cham: Springer International Publishing, 2021, pp. 606–617.

[12] M. Daily, S. Medasani, R. Behringer, and M. Trivedi, "Self-driving cars," *Computer*, vol. 50, no. 12, pp. 18–23, 2017.

[13] Arduino. (2021) Arduino platform. [Online]. Available: https://www.arduino.cc/

[14] M. Daadoo, S. Tarapiah, and S. Atalla, "Analysis and performance of a low cost multiple alarm security system for smart home based on gsm technology and controlling based on android smartphone," *European Journal of Scientific Research*, vol. 143, pp. 136–164, 12 2016.

[15] V. A. Zhmud, N. O. Kondratiev, K. A. Kuznetsov, V. G. Trubin, and L. V. Dimitrov, "Application of ultrasonic sensor for measuring distances in robotics," *Journal of Physics: Conference Series*, vol. 1015, p. 032189, may 2018. [Online]. Available: https://doi.org/10.1088/1742-6596/1015/3/032189

[16] A. Datasheet. (2021) L293b datasheet (pdf) - stmicroelectronics. [Online]. Available: https://pdf1.alldatasheet.com/datasheet-pdf/view/22430/STMICROELECTRONICS/L293B.html

[17] E. Tutorials. (2021) Dc motor. [Online]. Available: https://www.electronics-tutorials.ws/io/io_7.html

[18] J. Zou, Y. Han, and S.-S. So, *Overview of Artificial Neural Networks*. Totowa, NJ: Humana Press, 2009, pp. 14–22. [Online]. Available: https://doi.org/10.1007/978-1-60327-101-1_2

[19] Z. Waszczyszyn, "Fundamentals of artificial neural networks," in *Neural Networks in the Analysis and Design of Structures*, Z. Waszczyszyn, Ed. Vienna: Springer Vienna, 1999, pp. 1–51.

[20] R. S. Sutton, A. G. Barto, and R. J. Williams, "Reinforcement learning is direct adaptive optimal control," *IEEE Control Systems Magazine*, vol. 12, no. 2, pp. 19–22, April 1992.

[21] A. Dongare, R. Kharde, and A. D. Kachare, "Introduction to artificial neural network," *International Journal of Engineering and Innovative Technology (IJEIT)*, vol. 2, no. 1, pp. 189–194, 2012.

[22] R. S. Sutton and A. G. Barto, *Reinforcement learning: An introduction*. MIT press, 2018.

[23] A. Abraham, *Artificial Neural Networks*. American Cancer Society, 2005, ch. 129. [Online]. Available: https://onlinelibrary.wiley.com/doi/abs/10.1002/0471497398.mm421

[24] M. Buscema, "Back propagation neural networks," *Substance Use & Misuse*, vol. 33, no. 2, pp. 233–270, 1998. [Online]. Available: https://doi.org/10.3109/10826089809115863

[25] S. Sharma, S. Sharma, and A. Athaiya, "Activation functions in neural networks," *International Journal of Engineering Applied Sciences and Technology*, vol. 2, 2020. [Online]. Available: https://www.ijeast.com/papers/310-316,Tesma412,IJEAST.pdf

[26] R. Rojas, *The Backpropagation Algorithm*. Berlin, Heidelberg: Springer Berlin Heidelberg, 1996, pp. 149–182. [Online]. Available: https://doi.org/10.1007/978-3-642-61068-4_7

[27] J. McGonagle, G. Shaikouski, C. Williams, A. Hsu, J. Khim, and A. Miller. (2021) Backpropagation. [Online]. Available: https://brilliant.org/wiki/backpropagation/#formal-definition

[28] J. Brownlee, "What is the difference between a batch and an epoch in a neural network?" *Deep Learning; Machine Learning Mastery: Vermont, VIC, Australia*, 2018.

[29] J. Heaton, "Introduction to the math of neural networks (beta-1)," *Heaton Research Inc*, 2011.

[30] H. Demuth, M. Beale, and M. Hagan, *Neural network toolbox*. Mathworks, 1994.

[31] H. Jabbar and R. Z. Khan, "Methods to avoid over-fitting and under-fitting in supervised machine learning (comparative study)," *Computer Science, Communication and Instrumentation Devices*, pp. 163–172, 2015.

[32] T. Dietterich, "Overfitting and undercomputing in machine learning," *ACM computing surveys (CSUR)*, vol. 27, no. 3, pp. 326–327, 1995.

[33] M. van Otterlo and M. Wiering, *Reinforcement Learning and Markov Decision Processes*. Berlin, Heidelberg: Springer Berlin Heidelberg, 2012, pp. 3–42. [Online]. Available: https://doi.org/10.1007/978-3-642-27645-3_1

[34] R. E. Bellman and S. E. Dreyfus, *Applied dynamic programming*. Princeton university press, 1962.

[35] M. L. Puterman, "Chapter 8 markov decision processes," in *Stochastic Models*, ser. Handbooks in Operations Research and Management Science. Elsevier, 1990, vol. 2, pp. 331 – 434. [Online]. Available: http://www.sciencedirect.com/science/article/pii/S0927050705801720

[36] R. Brooks, "A robust layered control system for a mobile robot," *IEEE Journal on Robotics and Automation*, vol. 2, no. 1, pp. 14–23, March 1986.

[37] M. A. Kareem Jaradat, M. Al-Rousan, and L. Quadan, "Reinforcement based mobile robot navigation in dynamic environment," *Robotics and Computer-Integrated Manufacturing*, vol. 27, no. 1, pp. 135 – 149, 2011. [Online]. Available: http://www.sciencedirect.com/science/article/pii/S0736584510000700

[38] A. Medina-Santiago, J. Camas-Anzueto, J. Vazquez-Feijoo, H. Hernández-de León, and R. Mota-Grajales, "Neural control system in obstacle avoidance in mobile robots using ultrasonic sensors," *Journal of Applied Research and*

Technology, vol. 12, no. 1, pp. 104 – 110, 2014. [Online]. Available: http://www.sciencedirect.com/science/article/pii/S1665642314716104

[39] B. Markoski, S. Vukosavljev, D. Kukolj, and S. Pletl, "Mobile robot control using self-learning neural network," in *2009 7th International Symposium on Intelligent Systems and Informatics*, Sep. 2009, pp. 45–48.

[40] S. X. Yang and M. Meng, "Neural network approaches to dynamic collision-free trajectory generation," *IEEE Transactions on Systems, Man, and Cybernetics, Part B (Cybernetics)*, vol. 31, no. 3, pp. 302–318, June 2001.

[41] Pu Shi and Yujie Cui, "Dynamic path planning for mobile robot based on genetic algorithm in unknown environment," in *2010 Chinese Control and Decision Conference*, May 2010, pp. 4325–4329.

[42] X. Kang, Y. Yue, D. Li, and C. Maple, "Genetic algorithm based solution to dead-end problems in robot navigation," *International Journal of Computer Ajpplications in Technology*, vol. 41, no. 3-4, pp. 177–184, 2011. [Online]. Available: https://www.inderscienceonline.com/doi/abs/10.1504/IJCAT.2011.042693

[43] R. Abiyev, D. Ibrahim, and B. Erin, "Navigation of mobile robots in the presence of obstacles," *Advances in Engineering Software*, vol. 41, no. 10, pp. 1179 – 1186, 2010. [Online]. Available: http://www.sciencedirect.com/science/article/pii/S0965997810001018

[44] M. Duguleana and G. Mogan, "Neural networks based reinforcement learning for mobile robots obstacle avoidance," *Expert Systems with Applications*, vol. 62, pp. 104 – 115, 2016. [Online]. Available: http://www.sciencedirect.com/science/article/pii/S0957417416303001

[45] H. Xiao, Li Liao, and F. Zhou, "Mobile robot path planning based on q-ann," in *2007 IEEE International Conference on Automation and Logistics*, Aug 2007, pp. 2650–2654.

[46] O. Motlagh, D. Nakhaeinia, S. H. Tang, B. Karasfi, and W. Khaksar, "Automatic navigation of mobile robots in unknown environments," *Neural Computing and Applications*, vol. 24, no. 7-8, pp. 1569–1581, 2014.

[47] C.-T. Yen and M.-F. Cheng, "A study of fuzzy control with ant colony algorithm used in mobile robot for shortest path planning and obstacle avoidance," *Microsystem Technologies*, vol. 24, no. 1, pp. 125–135, 2018.

[48] J. K. Pothal and D. R. Parhi, "Navigation of multiple mobile robots in a highly clutter terrains using adaptive neuro-fuzzy inference system," *Robotics and Autonomous Systems*, vol. 72, pp. 48 – 58, 2015. [Online]. Available: http://www.sciencedirect.com/science/article/pii/S0921889015000895

[49] S. Gupta. (2018) Pull up and pull down resistor. [Online]. Available: https://circuitdigest.com/tutorial/pull-up-and-pull-down-resistor

[50] Arduino. (2021) Arduino ide. [Online]. Available: https://www.arduino.cc/en/software