



People's Democratic Republic of Algeria

Ministry Of Higher Education And Scientific Research

University Of Ghardaia

Faculty of Science and Technology

Department of Automatics and Electromechanics

N° d'ordre :
N° de série :

A thesis submitted in fulfillment of the requirements for the degree of

MASTER

Domain: *Sciences and Technologies*

Branch: *Automatics*

Specialty: *Automation and systems*

Theme

**Robotic Automated Scorpion and Snake Venom
Extraction System**

By :

SEKAYAR Soundous

Defended publicly on 30/06/2024

In front of the jury:

CHOUIA Faycal	MCB	Univ. Ghardaïa	President
MOSBAH Said	MCB	Univ. Ghardaïa	Examiner
HACENE Nacer	MCA	Univ. Ghardaïa	Supervisor

Academic year 2023/2024

بِسْمِ اللَّهِ الرَّحْمَنِ الرَّحِيمِ




Acknowledgment

Infinite praises and thanks be to Almighty Allah, the Most Gracious and Most Merciful, who endowed us with the capabilities and aided us in completing this modest endeavor.

I am profoundly grateful to my advisor, HACENE Nacer, for his unwavering support and insightful guidance throughout this project. His encouragement and mentorship have been invaluable and it has been a privilege to work under his supervision.

I extend my heartfelt thanks to our professors, academic staff, and personnel at the University of Ghardaia.

Above all, I am deeply thankful to my family for their relentless support whenever I needed it, always being there for me and guiding me through life's journey. I am immensely grateful for their unconditional love which continually inspires me to reach greater heights.



Abstract

This thesis aims primarily to design a robot capable of collecting the venom of scorpions and snakes. The robot is designed to operate autonomously within a scorpion or snake farm to ensure safety and efficiency. The robot consists mainly of an open-source Arduino programming card, motors, and ultrasonic sensors or a camera for greater accuracy. Additionally, the robot's external structure is covered with a flexible material that retains the venom. The robot identifies the scorpion or snake in the closed farm and pursues it, leading to its natural response of stinging and emptying the venom into the container.

Résumé

Ce mémoire vise principalement à concevoir un robot capable de recueillir les venins des scorpions et des serpents. Le robot est conçu pour fonctionner de manière autonome à l'intérieur d'une ferme à scorpions ou à serpents afin d'assurer la sécurité et l'efficacité. Le robot est constitué d'une carte programmable open-source Arduino, de moteurs et de capteur à ultrasons ou d'une caméra pour une précision accrue. De plus, la structure externe du robot est recouverte d'un matériau flexible qui retient le venin. Le robot identifie le scorpion ou le serpent dans la ferme fermée et le poursuit, ce qui entraîne sa réponse naturelle de piqûre et le vidage du venin dans le conteneur.

ملخص

تهدف هاته المذكرة بشكل رئيسي إلى تصميم روبوت يُمكن من الحصول على سموم العقارب و الأفاعي , تم تصميم الروبوت ليعمل بحركة ذاتية داخل مزرعة للعقارب أو الأفاعي لضمان السلامة والكفاءة , يتكون الروبوت بشكل اساسي من بطاقة برمجية مفتوحة المصدر أروينو و محركات و كاميرا للتحديد بدقة , كما يتم تغطية الهيكل الخارجي للروبوت بمادة مرنة حافظة للسم , يعمل الروبوت على تحديد العقرب أو الثعبان في المزرعة المغلقة و ملاحقته مما يؤدي إلى استجابة الطبيعية باللسع وبالتالي إفراغ السم في الحاوية .

Table of Contents

Theme	I
•	I
Acknowledgment	III
Abstract	IV
Table of Contents	V
List of Figures	IX
List of Tables	XI
List of Abbreviations	XII
Introduction	1
• Project Objectives	1
• Organization of the Thesis	1
Overview of Scorpions and Snakes in Algeria	4
1.1 Introduction	4
1.2 Types of scorpion in Algerian	4
1.2.1 Androctonus aeneas Koch, 1839	5
1.2.2 Androctonus amoreuxi (Audouin, 1826)	5
1.2.3 Androctonus australis (Linnaeus, 1758)	6
1.2.4 Buthus tunetanus (Herbst, 1800)	7
1.2.5 Leiurus hoggarensis	7
1.3 Types of snakes in Algeria	8
1.3.1 Egyptian Cobra (Naja Haje)	8
1.3.2 Sahara Sand Viper (Cerastes vipera)	8
.....	9

1.3.3 The Desert Horned Viper (Cerastes cerastes)	9
1.4 Statistics of scorpions and snakes bites in Algeria :.....	9
1.5 Scorpion and Snake Venoms and Promising Treatments :	11
1.6 Methods of Extracting Venom from Scorpions and Snakes :	13
1.6.1 Traditional Methods:	13
1.6.1.1 Methods for extracting venom from Scorpion:	13
1.6.1.2 Methods for extracting venom from snakes:	14
1.6.2 New Technical Methods :	14
1.6.2.1 The Snake Venom Milker	14
1.6.2.2 The Moroccan Automated Scorpion Venom Extraction System (VES Pro)	15
1.6.2.3 The Bridge Method	15
1.7 Conclusion.....	16
AI-Powered Object Detection Robots: Hardware & Software	18
2.1 Introduction :	18
2.2 Introduction to AI Techniques for Object Detection	18
2.3 Machine Learning and Deep Learning.....	19
2.4 Components of an Object Detection System.....	20
2.4.1 Hardware and Sensors: Cameras and Processors	20
2.4.1.1 Arduino.....	20
2.4.1.2 Raspberry pi- Raspberry pi camera.....	21
2.4.1.3 Pixycam	22
2.4.1.4 Esp 32 Cam:	24
2.4.2 Software and Algorithms	26
2.4.2.1 Collection of Data from various data source	26
2.4.2.2 Data Cleaning and Feature Engineering.....	26
2.4.2.3 Model building and Selection of ML Algorithm	27
2.4.2.4 Model Evaluation	27

2.4.2.5 Model Deployment.....	27
2.4.2.6 Applications of Object Detection Robots	27
2.5 Conclusion.....	27
.....	28
Hardware and Software System Design.....	29
3.1 Introduction.....	29
3.2 The First Method.....	29
3.2.1 Components	29
3.2.1.1 Pixycam.....	29
3.2.1.2 Arduino uno	30
*	31
L29	31
3.2.1.3 Motor Driver L298N	31
3.2.1.4 DC Motors.....	32
3.2.2 Circuit.....	34
3.2.3 Program and Speed Control	34
3.2.3.1 Speed without control.....	34
3.2.3.2 PID Speed Control	36
3.2.3.3 Fuzzy Logic Speed Control.....	38
3.3 The second method :	40
3.3.1 Components.....	40
3.3.2 Circuit.....	41
3.3.3 ESP32CAM Training	41
3.3.3.1 Collection of Data	41
3.3.3.2 Data Cleaning.....	42
3.3.3.3 Model building	43
3.3.3.4 Model Evaluation	45
3.3.3.5 Model Deployment.....	45

3.3.4 Program	46
3.4 Shape and external design	46
3.5 Conclusion.....	46
.....	47
Conclusion.....	48
References	49
Appendix A – Source Code(Pixycam)	52
Appendix B – Source Code(PID-Pixycam)	56
Appendix C – Source Code (Fuzzy Logic-Pixycam).....	61
Appendix D – Source Code(ESP32 Cam Program).....	70
Appendix E – Source Code(ESP32 –Arduino-PID)	83

List of Figures

Figure 1. 1 Androctonus aeneas	5
Figure 1. 2 Androctonus amoreuxi.....	6
Figure 1. 3 Androctonus australis	6
Figure 1. 4 Buthus tunetanus.....	7
Figure 1. 5 Leiurus hoggarensis	7
Figure 1. 6 Egyptian Cobra	8
Figure 1. 7 Sahara Sand Viper	9
Figure 1. 8 The Desert Horned Viper.....	9
Figure 1. 9 evolution of the number of scorpion stings, incidence rate, deaths, and lethality in Algeria.....	10
Figure 1. 10 Global incidence of snake bites	11
Figure 1. 11 Potential Therapeutic Uses of Scorpion Venoms	12
Figure 1. 12 Potential Therapeutic Uses of Snake Venoms.....	12
Figure 1. 13 Snake Venom Milker	15
Figure 2. 1 The development of objects detection technology [21]	19
Figure 2. 2 Machine Learning	19
Figure 2. 3 Deep Learning.....	20
Figure 2. 4 Aruino Hardware	21
Figure 2. 5 Raspberry Pi Board Ports.....	22
Figure 2. 6 Pixycam	23
Figure 2. 7 PixyMon application.....	23
Figure 2. 8 PixyMon application interface.....	24
Figure 2. 9 ESP32CAM-MB	25
Figure 2. 10 FTDI adapter.....	25
Figure 2. 11 ML Steps.....	26
Figure 3. 1 Arduino R3	30
Figure 3. 2 Motor Driver L298N.....	32
Figure 3. 3 Operating principle of the H-bridge.....	32
Figure 3. 4 TT DC Gearbox Motors.....	33

Figure 3. 5 The Robot circuit using Pixycam.....	34
Figure 3. 6 Flowchart of the robot without control	35
Figure 3. 7 The PID controller	36
Figure 3. 8 Flowchart of the robot With PID control.....	37
Figure 3. 9 General structure of a fuzzy controller	38
Figure 3. 10 Error and the Error Derivative Fuzzification.....	39
Figure 3. 11 Speed Fuzzification.....	39
Figure 3. 12 The Robot circuit using ESP32-Cam.....	41
Figure 3. 13 Training and Testing data	42
Figure 3. 14 Scorpions identification	42
Figure 3. 15 Initial model construction	43
Figure 3. 16 Feature Explorer	44
Figure 3. 17 Neural Network Settings.....	44
Figure 3. 18 Model Evaluation.....	45
Figure 3. 19 The Robot external design	46

List of Tables

Table 1 PixyCam ver. 2 CMUcam5 Specification	30
Table 2 Arduino Uno R3 Features	31
Table 3 TT DC Gearbox Motors characteristics	33
Table 4 PID controller gains.....	38
Table 5 Inference Table	40
Table 6 Model characteristics.....	45

List of Abbreviations

AI: Artificial Intelligence

CNNs: Convolutional Neural Networks

R-CNNs: Region-based Convolutional Neural Networks

YOLO: You Only Look Once

SSD: Single Shot MultiBox Detector

ML: Machine Learning

DL : Deep Learning

DSSD: Deconvolutional Single Shot Detector

IDE: Integrated Development Environment

ARM: Advanced RISC Machines

TTL: Transistor-Transistor Logic

HDMI : High-Definition Multimedia Interface

IOT: Internet of Things

FTDI: Future Technology Devices Internationa

COCO: Common Objects in Context

USB: Universal Serial Bus

FOMO: Fast Object More Object

FPGAs: Field-Programmable Gate Arrays



Introduction

Introduction

Breeding farms, milking and extracting scorpion and snake venom are among the most dangerous and expensive medicines in the world

Worldwide approximately 5.4 million people are estimated to suffer from snake bites annually, resulting in 81,410 to 137,880 deaths each year. It also causes about three times as many amputations and other permanent disabilities each year [1]. The annual number of scorpion stings exceeds 1.2 million, leading to over 3,250 deaths globally [2].

Neurotoxins present in venom hold medical significance, serving as the driving force in the field of scorpion and snake venom research. Venom milking or extraction has been the primary step to obtain that vital material

There are numerous methods for extracting toxins from scorpions and snakes, but they are laden with risks, whether to the individuals conducting the process or to the creatures themselves.

In this project we will mainly work on how to create and program a robot to get venom safely using Arduino Uno and PixyCam

- **Project Objectives**

The primary goal of this project is to develop an automatic system with a self-propelled robot that enables us to obtain scorpion and snake venom without handling these dangerous creatures or even putting them in danger. This will be achieved using an Arduino Uno board and a detection camera. An ultrasonic sensor can also be used instead of a camera in the simple model.

- **Organization of the Thesis**

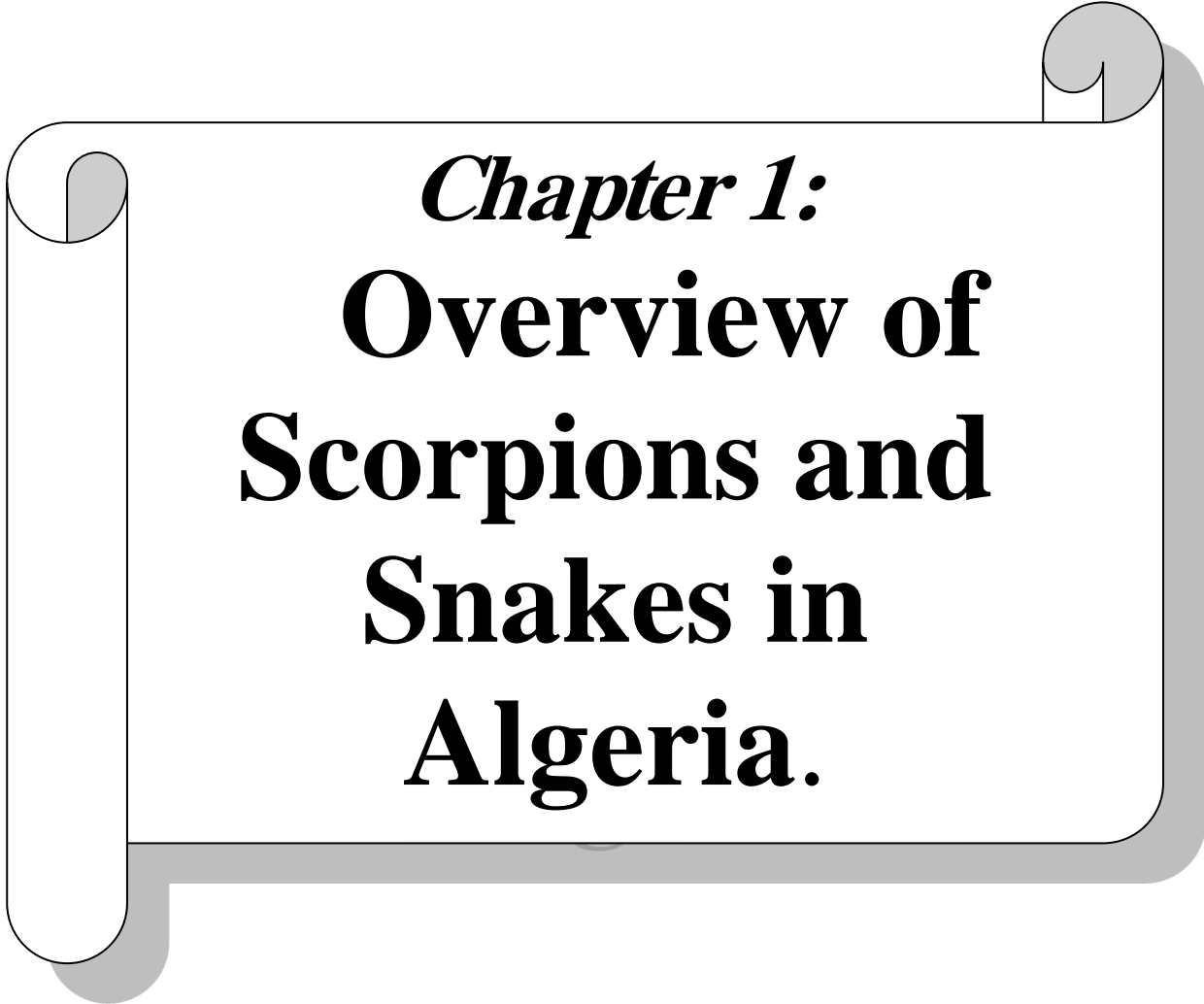
The thesis is organized as follows:

Chapter 1 provides an overview of scorpions and snakes in Algeria, their large populations, and both modern and traditional methods of extracting their venoms. It also discusses the importance of these venoms and future therapeutic prospects.

Chapter 2 introduces the concept of artificial intelligence and machine learning, along with the hardware and software used in commonly deployed tracking robots, and explores common applications of tracking robots. This chapter serves as an introduction to the core of the presented work.

Chapter 3, the final chapter, presents system design using two different approaches: artificial intelligence and other techniques, along with flowcharts of algorithms used in the tracking robots we have developed. It also covers motion control methods using two types of controllers. and the using of artificial intelligence techniques (CNN) for camera training.

In conclusion, a general summary of the work is presented followed by a list of references.



Chapter 1:
**Overview of
Scorpions and
Snakes in
Algeria.**

Chapter 1

Overview of Scorpions and Snakes in Algeria

1.1 Introduction

Algeria boasts a vast biological wealth, including 46 species of scorpions and 31 species of snakes. Because the venoms of these creatures hold promise for treating various intractable diseases, they represent liquid gold that can be a genuine source of wealth and raw material for exploitation in Algeria.

1.2 Types of scorpion in Algerian

According to recent studies of old scorpion materials preserved at the National Museum of Natural History in France, as well as materials collected recently in Algeria, it has been found that Algeria encompasses 46 species and subspecies of scorpions distributed among 14 genera and three different families (Buthidae, Euscorpiidae, and Scorpionidae) [3].

Algeria hosts at least 26 species of endemic species, representing 58% of its national richness. Meanwhile, in Egypt, local distribution represents only 17% (6 species) [04].

Considering the list of scorpions discovered in the last two decades in Algeria (24 species), it is estimated that it is still far from being comprehensive, given the possibility of finding other species. Consequently, it is estimated that the diversity of scorpions in Algeria currently represents only 70% of the actual number that may exist in Algeria. All scorpions are venomous, but not all of them are lethal to humans. The proportion of scorpions that pose a danger to humans in Algeria is estimated at 20%, from the families: Androctonus, Buthiscus, Buthus, and Leiurus [3].

These dangerous species include 9 types that are medically significant, with the 4 most dangerous species being as follows:

1.2.1 *Androctonus aeneas* Koch, 1839

This dark brown to black scorpion (**Figure 1.1**), belonging to the Buthidae family and *Androctonus* Ehrenberg genus (1828), is found in the central horizontal range of Algeria from Tébessa and Khenchela in the east to Naâma in the west, as well as in the Algerian steppes (M'Sila, Sidi Bel Abbès) and even in the desert regions of El Oued and Ghardaïa. This species can reach a length of 8 cm, with a lighter tip, movable legs, and pincers [3] . It is classified as the largest black species in Algeria [3] . Although the toxicity of this species has not been tested, it is suspected to be dangerous to humans



Figure 1. 1 *Androctonus aeneas*

1.2.2 *Androctonus amoreuxi* (Audouin, 1826)

This scorpion also belongs to the Buthidae family and the genus *Androctonus* Ehrenberg (1828). It is widely distributed in Algeria in large numbers and is considered one of the largest scorpion species in Algeria [3] , reaching a length of up to 12 cm (**Figure 1.2**) . Despite being less dangerous compared to other species of the genus *Androctonus*, several cases of stings are recorded annually with this species, especially in El Oued and Ghardaïa.



Figure 1. 2 *Androctonus amoreuxi*

1.2.3 *Androctonus australis* (Linnaeus, 1758)

It is also considered one of the largest species in Algeria, reaching a length of 10 cm (**Figure 1.3**). It is the most widespread in the northern Sahara and is commonly found near residential areas. This species is considered the most dangerous globally and is responsible for a high mortality rate.



Figure 1. 3 *Androctonus australis*

1.2.4 *Buthus tunetanus* (Herbst, 1800)

This medium-sized species (5 to 7 cm) belongs to the *Buthus occitanus* complex (**Figure 1.4**), which includes 8 species found in Algeria with varying degrees of danger. This species was previously ranked second after *Androctonus australis* in terms of morbidity [3].



Figure 1. 4 *Buthus tunetanus*

1.2.5 *Leiurus hoggarensis*

This species reaches a size of 9.5 cm and was recently discovered in Algeria (**Figure 1.5**). Like its predecessors, this species is considered dangerous to humans due to its belonging to the genus *Leiurus* [3].



Figure 1. 5 *Leiurus hoggarensis*

1.3 Types of snakes in Algeria

Algeria is home to 31 species of snakes [5], with at least 10 of them being venomous. These snakes range from desert-dwelling species to those found in more humid regions .

These venomous snakes hold significant medical importance in Algeria. The top 3 species are :

1.3.1 Egyptian Cobra (*Naja Haje*)

Also known as the Brown Snake, this highly venomous snake (**Figure 1.6**) is one of the largest snakes found in Africa [6]. It is present in several regions in Algeria, including the northeastern coastal area and the high plateau region.



Figure 1. 6 Egyptian Cobra

1.3.2 Sahara Sand Viper (*Cerastes vipera*)

The Sahara Sand Viper (**Figure 1.7**), also known as the Hornless Viper or Lesser Cerastes, is characterized by its small size and high venom potency. This snake is found in the deserts and arid regions of Algeria [7] .



Figure 1. 7 Sahara Sand Viper

1.3.3 The Desert Horned Viper (*Cerastes cerastes*)

This venomous snake (**Figure 1.8**) is one of the most widespread in the Algerian desert and is distinguished by the prominent horns above its eyes and its distinctive venom. According to writer Peter David Fraser, its toxins can even cause mental disorders [8].



Figure 1. 8 The Desert Horned Viper

1.4 Statistics of scorpions and snakes bites in Algeria :

In Algeria, as in many other parts of the world, scorpion and snake bites pose a significant health problem, especially in desert and rural areas. With the diverse and varied types of scorpions and snakes in Algeria, their impact and danger differ according to their species and

toxicity. However, most of these bites require urgent medical intervention, and in many cases, these bites can also lead to death.

In 2021, the number of scorpion stings in Algeria reached 40,127, resulting in 22 deaths [9]. (Figure 1.9) represents the yearly evolution of the number of scorpion stings, incidence rate, deaths, and lethality in Algeria .from 1991 to 2021 [9].

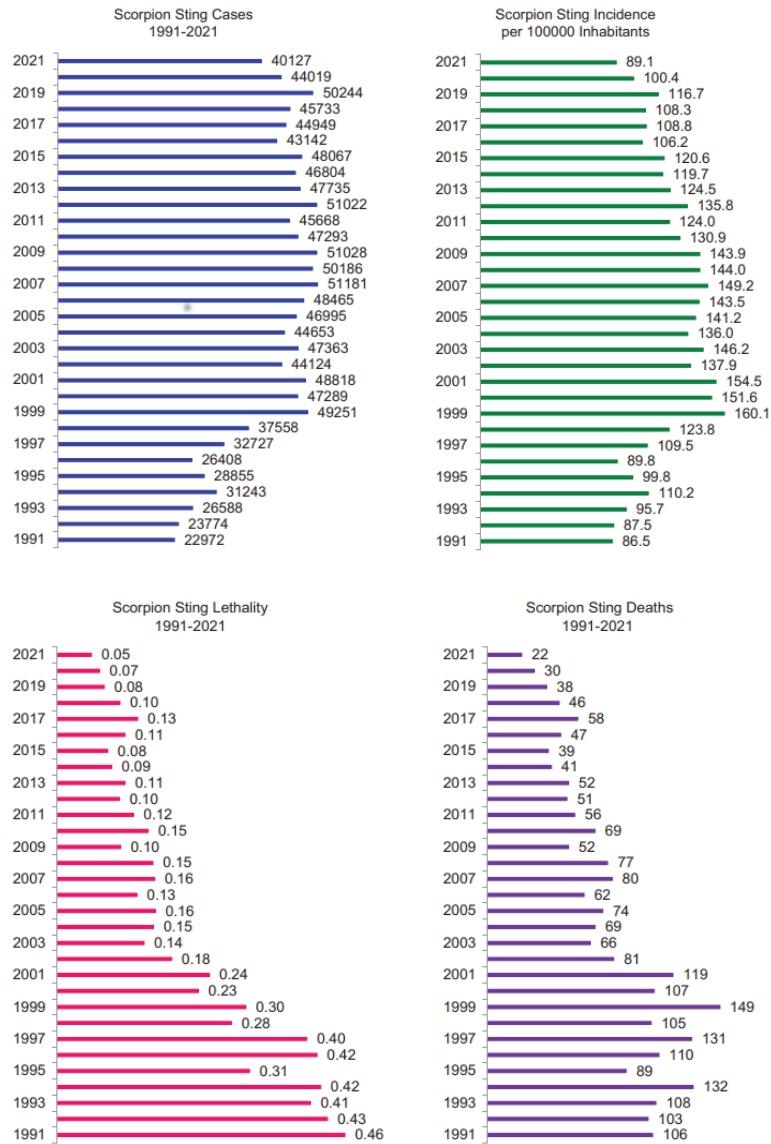


Figure 1. 9 evolution of the number of scorpion stings, incidence rate, deaths, and lethality in Algeria

On the other hand, the average number of deaths due to snake bites annually is about 25 cases[10]. In the (Figure 1.10) the number of snake bites worldwide in the year 2009 is shown. It illustrates that in the northern part of Algeria, there are between 10 to 100 bites per 100,000 individuals, while in the southern part of the country, the number exceeds 100 bites[11].

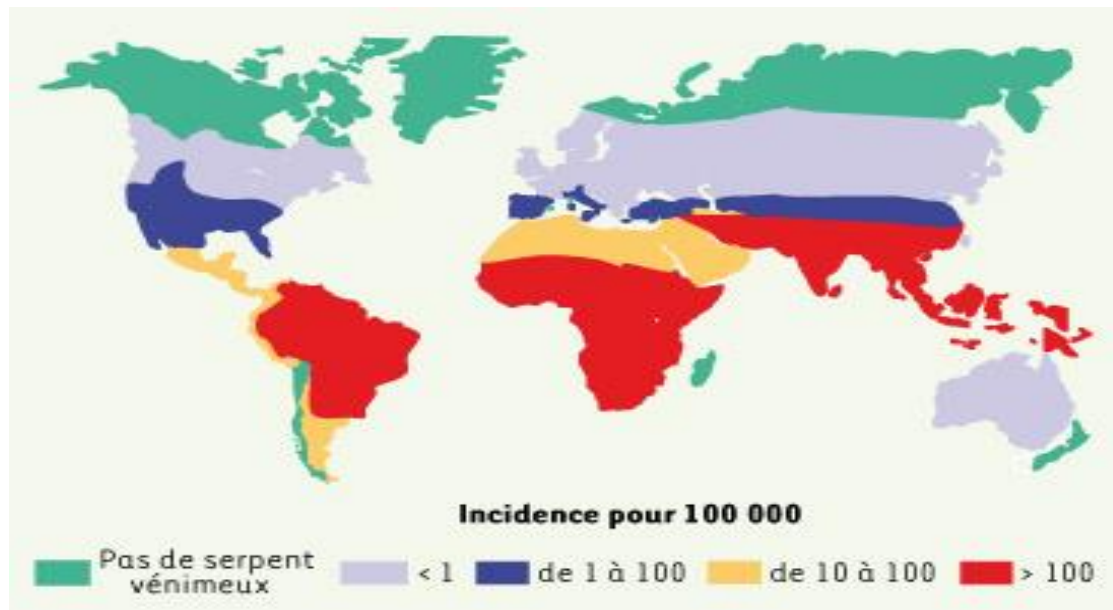


Figure 1. 10 Global incidence of snake bites

1.5 Scorpion and Snake Venoms and Promising Treatments :

The most severe harm that snake and scorpion stings can cause is death. However, in less severe cases, they can cause several distressing symptoms such as blood clotting or dissolution, release of pro-inflammatory cytokines, gastrointestinal symptoms, acute respiratory disorders, and organ damage that may necessitate amputation [12]. Nevertheless, research indicates that the peptides, amino acids, and organic compounds found in these venoms are promising candidates for future treatments [13],[14] .

The potential treatments using scorpion venoms summarized in **(Figure 1.11)** [13].

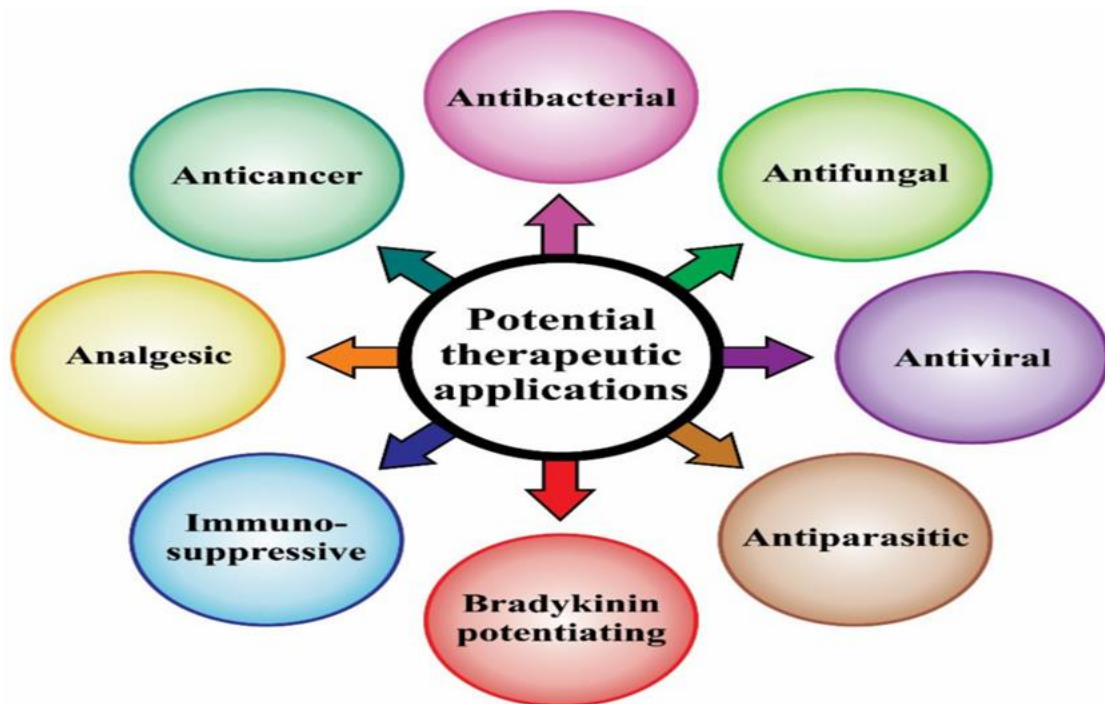


Figure 1. 11 Potential Therapeutic Uses of Scorpion Venoms

Similarly, snake venoms are involved in various treatments, which are summarized in the (Figure 1.12) [14] :

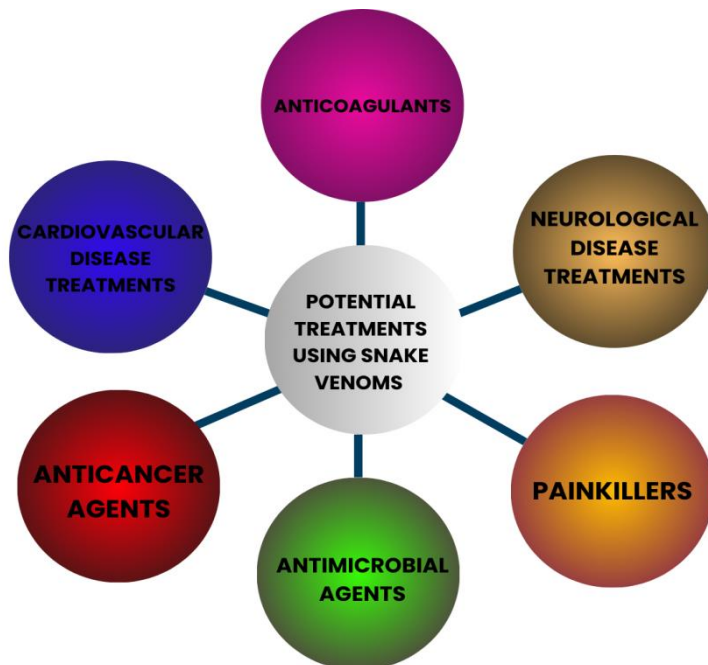


Figure 1. 12 Potential Therapeutic Uses of Snake Venoms

1.6 Methods of Extracting Venom from Scorpions and Snakes :

Given that scorpion and snake venoms are highly valuable biological substances, often sold for thousands of dollars, the commercial and investment prospects in this field are significant, whether for research or pharmaceutical industries. This has led to intense competition to innovate easier methods for extracting these substances for sale. Traditional methods, which are still used today, come with risks, while newer methods have been developed to minimize these risks. Therefore, these methods can be classified into two categories:

1.6.1 Traditional Methods:

These are the most common and widely used methods, relying entirely on human handling and interaction with these deadly creatures, whether scorpions or snakes.

1.6.1.1 Methods for extracting venom from Scorpion:

- **Manual Extraction Method**

This method is done by catching the scorpion and stimulating it manually to extract the venom. It is one of the oldest methods [15]

- **Electrical Extraction Method**

This method provides the best results compared to other traditional methods [15]. It depends on dipping the scorpion in a salt solution and then shocking the tail with a weak electric current to stimulate the gland to release the poison. [16]

- **Mechanical Stimulation Method**

This method relies on mechanically stimulating the scorpion's venom gland, using specialized tools such as needles and tweezers.

- **Maceration Method**

This method involves cutting and smashing part of the scorpion's tail (telson) to extract the venom.

- **Aspiration Method**

In this method, special suction tools are used to extract venom from the scorpion's glands.

1.6.1.2 Methods for extracting venom from snakes:

- **Method Without External Pressure**

In this method, the snake's fangs are inserted into a plastic covering on the glass without applying any external pressure on the venom glands. This method is not used for all types of snakes but is suitable for some specific types such as cobras and kraits

- **Method of Pressing the Venom Glands**

Just like the previous method, the snake's fangs are inserted into the milking cup, but in this method, gentle pressure is placed on the snake's venom glands to obtain the venom.

- **Electrical Stimulation Method**

This method involves allowing the snake to bite the milking cup and then applying a 6 volt electric shock. This technique is used to milk cobra snakes.

Corrected expression: These methods, while effective and widely used, are fraught with danger due to the necessity of full manual handling of these venomous creatures. Additionally, repeated electrocution of a scorpion's tail directly impacts its life expectancy. Scorpions subjected to electric shock extractions live an average of 4.46 years less than their counterparts [17].

1.6.2 New Technical Methods :

Since technology penetrates all fields, the pursuit of technological solutions has become imperative for all areas, including the field of toxin extraction. Specialists have strived to innovate solutions that can be obtained, and these solutions can be summarized as follows:

1.6.2.1 The Snake Venom Milker

The Snake Venom Milker is a device invented in the United States by Raymond D. Berger and patented in 1975. It can be explained based on the following Figure (**Figure 1.13**) where Device 11 consists of 12 rubber containers that are heated to lure the snake with prey. Container 12 is connected to the end of column 13 and handle 14, where column 13 is made of a heatable material and extends to electrical conductor wire 15 from a power source. Thus, enticing the snake involves some of the spherical containers and emptying venom into them [18].

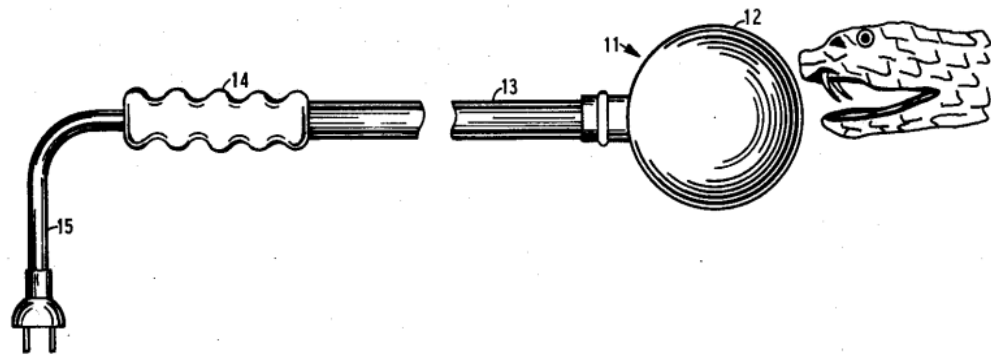


Figure 1. 13 Snake Venom Milker

1.6.2.2 The Moroccan Automated Scorpion Venom Extraction System (VES Pro)

It is an automated device created by Moroccan researchers at Hassan II University in Casablanca. The device works by fixing the scorpion, then delivering an electric shock to its tail to extract the venom, which is collected in a tube attached to the device. Thus, it's an automation of the Electrical extraction method [19].

1.6.2.3 The Bridge Method

This practical method, developed by the Algerian Youssef Ferme, revolves around placing two areas with a narrow bridge between them. Silicon balls are placed on the bridge as bait, while scorpions are placed in the first area. They are then stimulated by sound waves, causing the scorpions to move towards the bridge to cross to the second area. As they encounter the bait (silicon balls), they sting them and empty their venom into them. Through this method, venom extraction is successfully achieved .

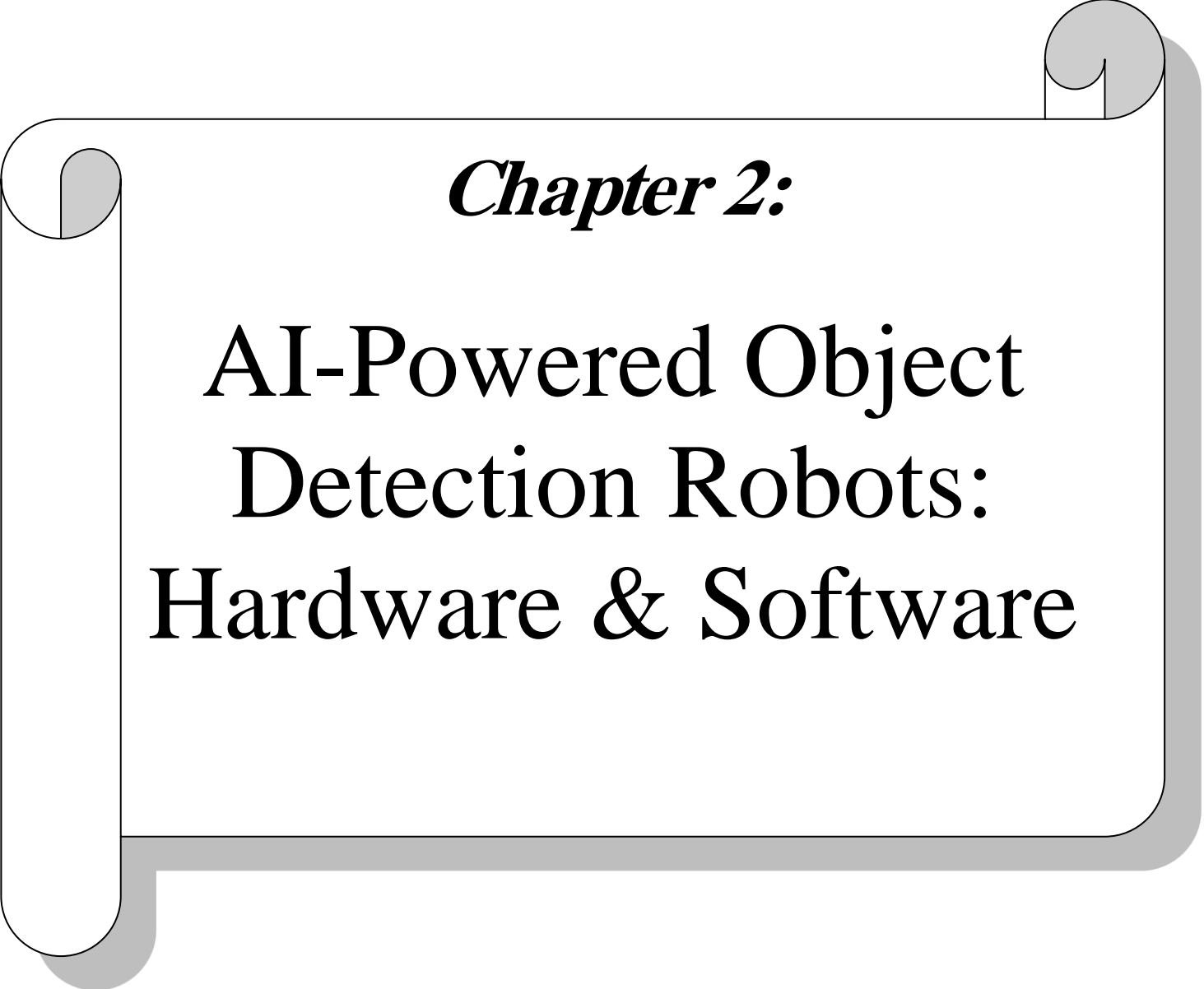
While these modern methods may ensure some degree of safety, they still suffer from some drawbacks that can be highlighted as follows:

- The danger of using the Snake Venom Milker, despite its patent, lies in its stick-like design, requiring the user to hold its handle, potentially making them appear as another prey to the snake instead of a silicon ball.

- The second device, as mentioned earlier, employs the same electric shock extraction method from the scorpion's tail but in an automatic manner, negatively affecting the scorpion's average lifespan.
- The final method may seem balanced in terms of safety for both the scorpions and their handlers. However, the issue arose when the inventor, after not achieving 100% results, admitted that some scorpions preferred to escape or hide without reacting to the baits, while others reacted by stinging and emptying venom into them

1.7 Conclusion

In this chapter, we mentioned the types of scorpions and snakes in Algeria , and we also discussed the number of recorded stings in previous years, as well as the potential treatments using their venoms. Finally, we covered the most important traditional and modern methods for extracting these venoms and the drawbacks of these methods.

A decorative graphic of a scroll with a grey shadow, framing the chapter title. The scroll has a white background and a black outline, with rounded corners and a small grey circle at the top right corner.

Chapter 2:

**AI-Powered Object
Detection Robots:
Hardware & Software**

Chapter 2

AI-Powered Object Detection Robots: Hardware & Software

2.1 Introduction :

Not long ago, the field of robotics, programming, and the use of artificial intelligence tools was exclusive to specialists. However, today this field has become accessible even to amateurs, thanks to the availability of various programmable boards and ready-made, AI-based, simplified software that allow amateurs to program them for various projects, primarily in object detection and tracking robots.

2.2 Introduction to AI Techniques for Object Detection

It is the task of analyzing images and videos to recognize objects and determine their locations within them. Recently, AI capabilities have been significantly enhanced by deep learning to improve its object detection and even classification capabilities (**Figure 2.1**) . Among the main techniques are [20] :

- Convolutional Neural Networks (CNNs),
- Region-based Convolutional Neural Networks (R-CNNs) (include Fast R-CNN, Faster R-CNN, and Mask R-CNN)
- The YOLO (You Only Look Once) system
- The Single Shot Multibox Detector (SSD), which is similar to YOLO.

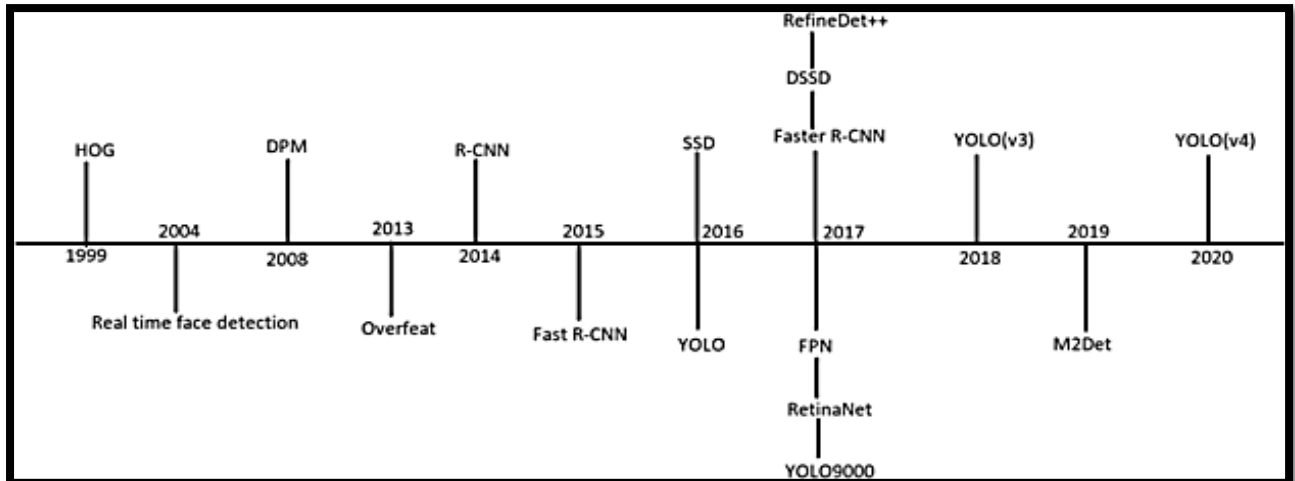


Figure 2. 1 The development of objects detection technology [21]

2.3 Machine Learning and Deep Learning

Artificial intelligence describes the state that allows a machine to mimic human intelligence, such as thinking, learning, and problem-solving. Machine learning (ML) (**Figure 2.2**) is considered a branch of artificial intelligence and focuses on building systems that learn from data. Its techniques include supervised learning, unsupervised learning, reinforcement learning, and deep learning (DL) (**Figure 2.3**), which relies on deep neural networks and is the most widely used in the field of object detection. And that is because it is the fastest and most accurate [21].

The main difference between traditional machine learning and deep learning is that in traditional machine learning, we perform two separate steps: first, we extract important features from the data, and then we classify these features. However, in deep learning, we do this in one step within the neural network, where the network extracts and classifies features directly [22].

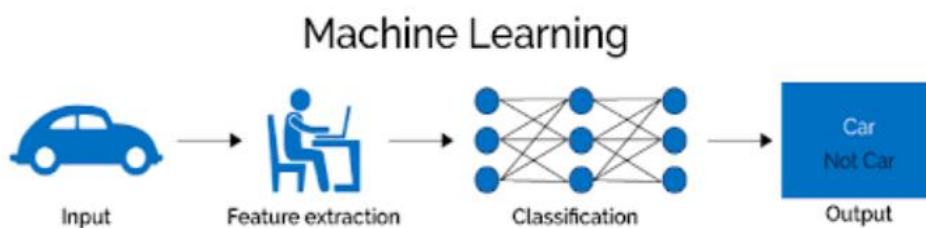


Figure 2. 2 Machine Learning

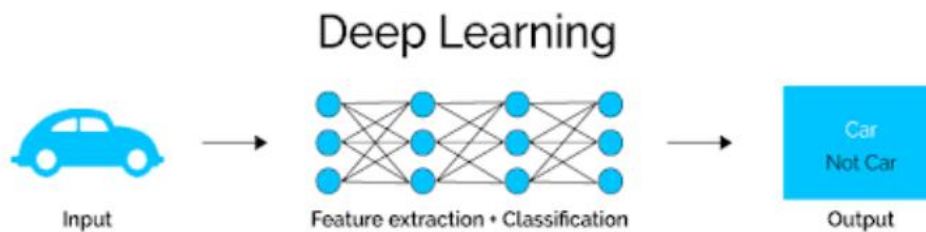


Figure 2. 3 Deep Learning

2.4 Components of an Object Detection System

with the help of open-source software, object detection systems and object tracking robots can be implemented using simple and readily available equipment.

2.4.1 Hardware and Sensors: Cameras and Processors

2.4.1.1 Arduino

Arduino is an integrated system or open-source environment used to interface software with the physical world. It relies on various-sized programmable boards that can be programmed and connected to a variety of sensors to implement both simple and advanced projects.

These boards are programmed and the programs are uploaded to them using its integrated environment (IDE) based on the Processing project, supporting programming languages like C and C++ [23].

These small control boards are primarily manufactured by Smart Projects in Italy, as well as by many other vendors, using 32-bit ARM processors or 8-bit Atmel AVR microcontrollers [23].

The Arduino board, in addition to the microcontroller, is equipped with a USB port, a power input, a power circuit, and input/output ports (**Figure 2.4**).

These boards can be connected to various types of sensors and transceivers, such as cameras like ESP32 and Pixycam. Arduino can be used to implement simple and uncomplicated object tracking robots due to the limited capabilities of these boards.

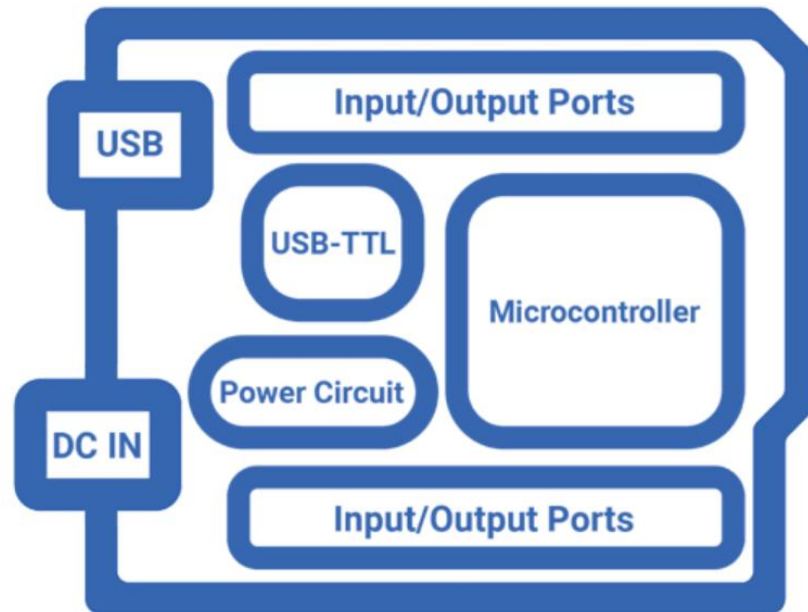


Figure 2. 4 Arduino Hardware

2.4.1.2 Raspberry pi- Raspberry pi camera

The Raspberry Pi board (**Figure 2.5**) is considered a computer device designed by Eben Upton in 2006 and developed by professors at the University of Cambridge. It gathers all the necessary components to run an operating system and use it as a real computer. This versatile card allows for facilitating [24]:

- computer programming learning
- democratizing the use of automation and robotics
- developing specific applications.

Raspberry Pi makes digital technologies usable by everyone so that everyone can solve problems or create new projects.

This small computer can be used when connected to the camera (Pi Camera) to run image and video processing algorithms and detect complex objects using frameworks such as TensorFlow Lite or OpenCV. It is also utilized in advanced software and various types of artificial intelligence algorithms. This board is the most popular and widely used, especially with its own camera, in object tracking robots using artificial intelligence because of its high capability in image processing and also because of its large capacity.

With its significant capability compared to Arduino, Raspberry Pi also comes with multiple ports such as HDMI and Ethernet, along with a dedicated port for audio output and another for connecting it to a specialized camera (Raspberry Pi camera), simplifying its usage and internet connectivity [25]. However, Arduino enjoys its simplicity, programming ease, and extremely low cost compared to Raspberry Pi.

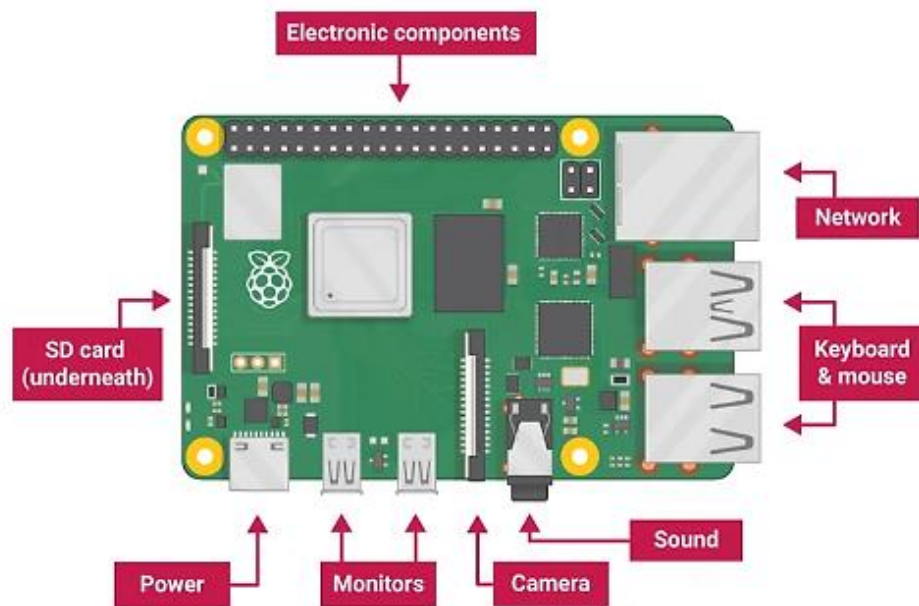


Figure 2. 5 Raspberry Pi Board Ports

2.4.1.3 Pixycam

This vision system is small, fast, and easy to use. It is also low-cost and open-source, available on Windows, Linux, and Mac. The system has a detection speed of 50 times per second and connects to Arduino using the included cable. It also works with Raspberry Pi and similar controllers and provides libraries for Arduino, Raspberry Pi, and other units. This camera supports C, C++, and Python programming languages [26].

This vision system comes with a camera and an accompanying cable for connecting to control units (**Figure 2.6**), compatible with the PixyMon application (**Figure 2.7**), which works with Windows, Linux, and Mac. The application displays real-time camera feed and helps adjust camera settings and identify objects. Simply connecting the camera to the computer via a USB cable is enough to start the application working and displaying the image [26].

The (**Figure 2.8**) illustrates the PixyMon application interface, where [26]:

- **Buttons:** These buttons represent the most common actions in PixyMon, conveniently located at the top of the main window :

- **Stop/Resume:** By clicking this button, the current program operation is halted. This is useful for capturing a frame or entering commands into the console window. Clicking this button again resumes program execution.
 - **Default Program:** Clicking this button executes the default program, which is the program that starts when Pixy2 powers up. The default program (by default!) is the color connected components program, but it can be configured.
 - **Raw Video:** Clicking this button displays unprocessed raw video.
 - **Configure:** Clicking this button opens the Configure Dialog, which contains various configurable parameters for Pixy2 and PixyMon.
- **Video Window:** This area displays various types of raw or processed video rendered by PixyMon.
 - **Status Bar:** This section displays current status messages.
 - **Frames Per Second:** As part of the status bar, the frames per second (fps) are displayed.
 - **Cursor Location:** Moving the mouse cursor across the video windows displays the image coordinates of the cursor.

Pixy employs a color-based filtering algorithm to detect objects, calculating the hue and saturation for each RGB pixel in the image [26]. Despite being a simple and user-friendly camera with all its software modules, the main issue with this camera is its susceptibility to lighting and exposure, as color filtering algorithms are sensitive to these changes, leading to a decrease in camera accuracy.



Figure 2. 6 Pixycam



Figure 2. 7 PixyMon application

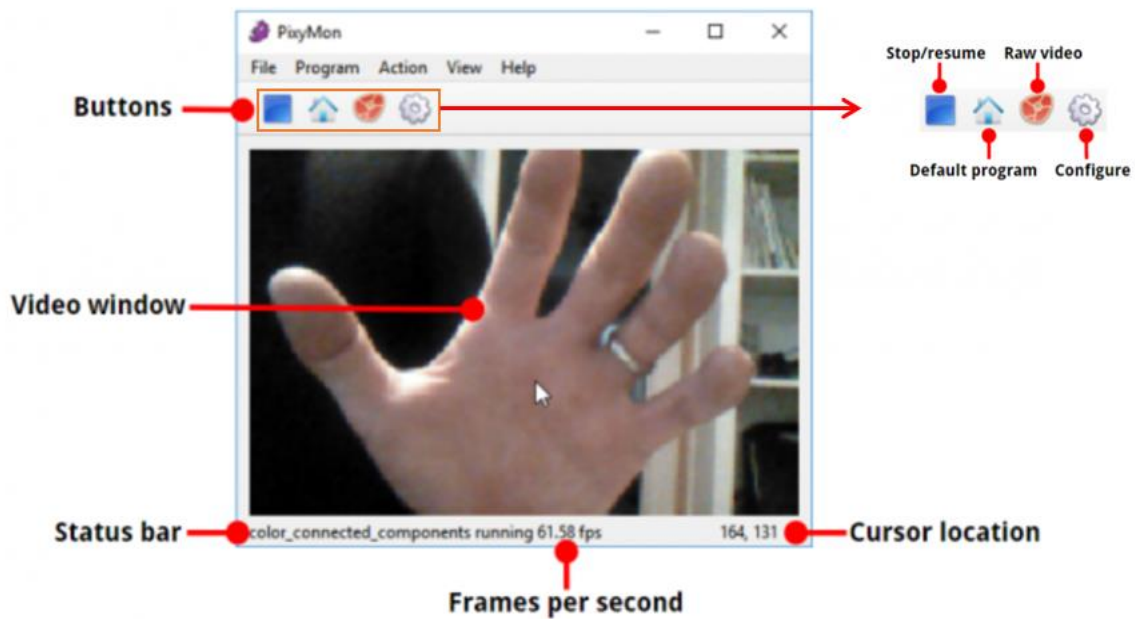


Figure 2. 8 PixyMon application interface

2.4.1.4 Esp 32 Cam:

The ESP32 CAM module is a low-cost microcontroller unit based on the ESP32 chip, featuring a compact OV2640 camera and a microSD card slot. This module integrates Bluetooth, WiFi, and BLE Beacon, powered by two high-performance 32-bit LX6 CPUs. The operating frequency can be adjusted between 80MHz and 240MHz. The architecture is designed with a pipeline structure and includes a Hall sensor, on-chip sensor, temperature sensor, and more. Manufactured by Espressif, this module is ideal for industrial wireless control, smart home devices, wireless monitoring, and IoT applications that require advanced camera functionalities such as image recognition and tracking [27].

The ESP32-CAM is programmed using the Arduino IDE, which provides an integrated development environment with ready-made libraries to easily connect and program this board. Additionally, AI libraries can be integrated into the ESP32-CAM for IoT applications, Object detection, and tracking.

There are several types of ESP32CAM, some of which can be connected directly to the development environment via a USB port (such as the ESP32-Eye), while others, like the ESP32 AI Thinker, require an interface to connect to the development environment. This can be done using :

- ESP32CAM-MB : The ESP32-CAM AI-Thinker MB programmer shield (**Figure 2.9**) is a shield that connects to the GPIO pins of the ESP32-CAM. The programmer comes with a USB chip to connect and program the ESP32-CAM. The shield is equipped with a reset button that used for start the program existen on the bord, and the button can also be used to activate the board's flash [28] .
- FTDI adapter : This USB to TTL serial adapter (**Figure 2.10**) works as an ideal solution for various applications, including programming firmware for microcontrollers, such as the ESP32-CAM [29].
- Arduino board : An Arduino board can also be used as an interface to upload programs to the ESP32-CAM.

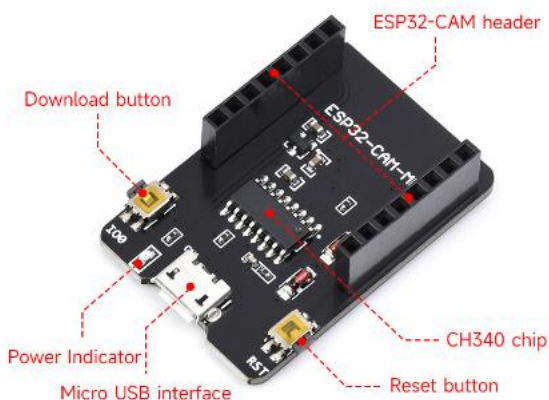


Figure 2. 9 ESP32CAM-MB

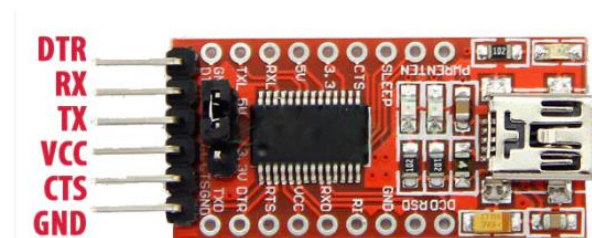


Figure 2. 10 FTDI adapter

2.4.2 Software and Algorithms

To implement AI-based object detection systems, one must first create and train these models using ML. This involves Five steps (Figure 2.11)[30] :

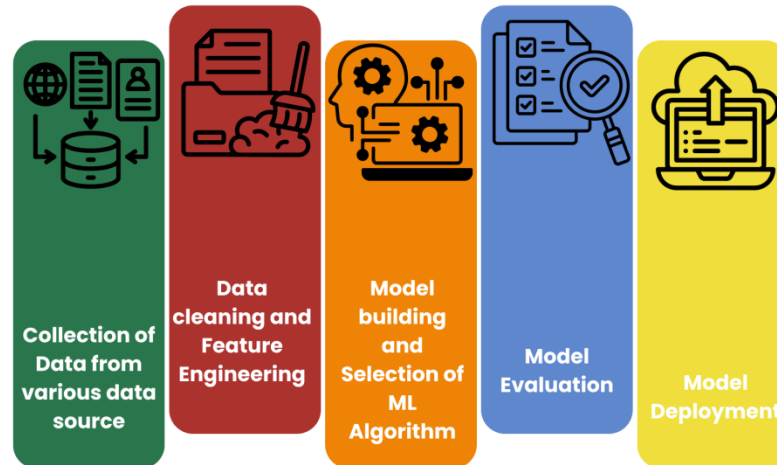


Figure 2. 11 ML Steps

2.4.2.1 Collection of Data from various data source

This is the first and most important step in the process of creating artificial intelligence models. It involves selecting Data for the model to train on .This process is subject to conditions, the most important of which is the clarity of the images to increase the model's learning accuracy. Data or images can be collected by capturing them or through available libraries online and databases like COCO and Kaggle.

The larger the amount of collected data, the more accurate the model training will be, but it will require more storage space.

2.4.2.2 Data Cleaning and Feature Engineering

This step is crucial for improving the performance of the model and training it optimally, during which [30]:

- Data cleaning: by improving unclear data and deleting incorrect data.
- Feature Engineering: identifying features from the data that facilitate the model training

2.4.2.3 Model building and Selection of ML Algorithm

Building the model and selecting the algorithm, depending on the problem to be solved, usually CNNs are used in object detection problems, and hyperparameters are also modified to improve the performance of the model (number of layers, Learning rate, etc.).

Then the model is trained using the data selected in the previous step through training and testing.

2.4.2.4 Model Evaluation

In this step, the model is evaluated according to various metrics, which are:

- Precision
- Recall
- F1 Score

2.4.2.5 Model Deployment

Machine learning models are deployed into a production environment, including microcontrollers (like Arduino), to be used as ready models for executing their trained algorithms in real-time. They are utilized for identification, classification, programs for tracking robots, or solving object detection problems.

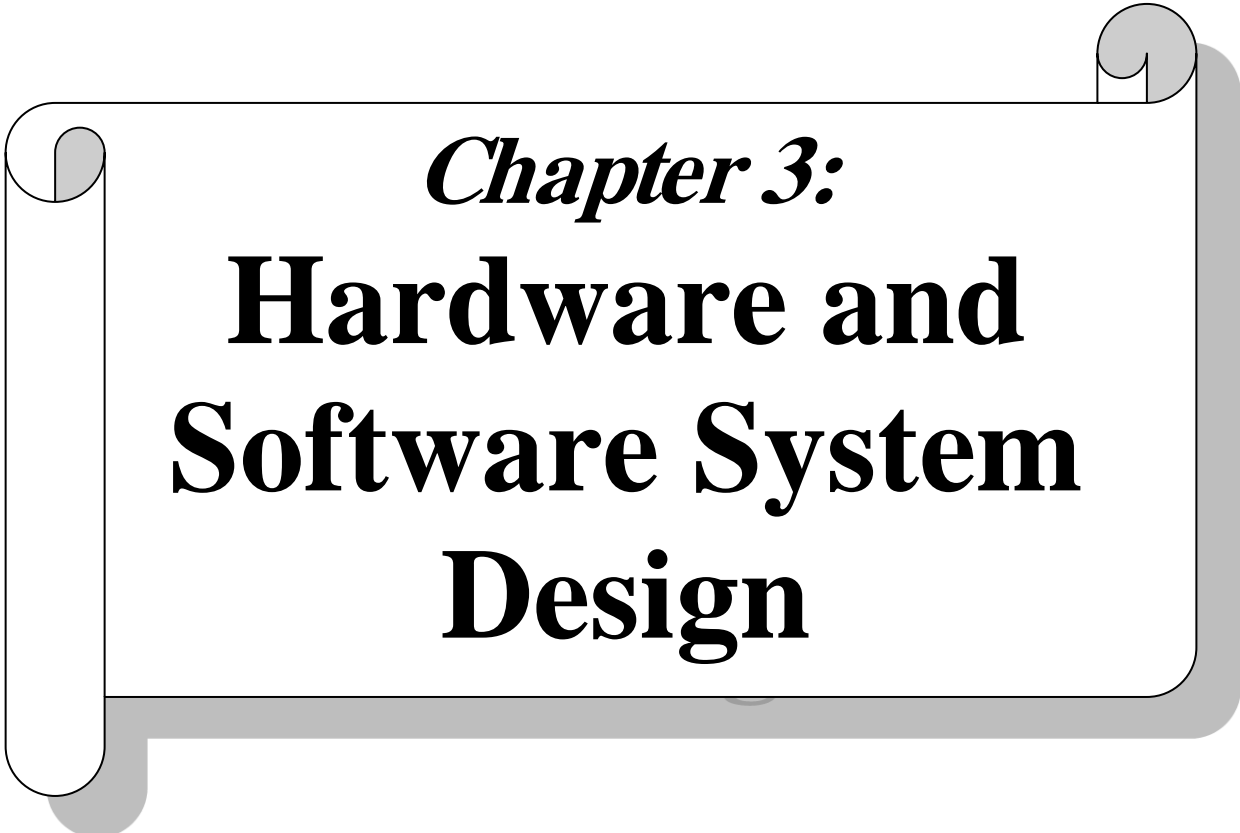
2.4.2.6 Applications of Object Detection Robots

The use of Object Detection Robots and object-tracking robots is sweeping across all areas of life in today's world, and here are some of its applications :

- Logistics and Autonomous Delivery [31]
- Autonomous robots in agriculture and plant care [32]
- Search and Rescue Robots [33]

2.5 Conclusion

In This chapter, we define the artificial intelligence techniques , Machine Learning and Hardware- Software used in object detection and tracking robots And also some uses of these robots.



Chapter 3:
**Hardware and
Software System
Design**

Chapter 3

Hardware and Software System Design

3.1 Introduction

We are working on developing a new type of robot capable of extracting venoms, which have been discussed for their significant importance and the actual wealth they represent, from both scorpions and snakes, using artificial intelligence techniques and modern technologies to track snakes and scorpions and pose a threat to them, thereby causing them to sting the robot and consequently obtain the poisons in a rubber container carried by the robot.

3.2 The First Method

In this case, we used a pre-programmed camera, PixyCam , and connected it to the Arduino. Then, we moved the robot according to the camera's vision to track the scorpion.

3.2.1 Components

3.2.1.1 Pixycam

We used the PixyCam ver. 2 CMUcam5 model with the Specification showing in the **Tabel 1** [34]:

CPU	2-core NXP LPC4330, 204 MHz
RAM	264 KB
Flash memory	2 MB
Image sensor	Aptina MT9M114, 1296 × 976 pixels
Viewing angles:	Horizontal: 60 ° Vertical: 40 °
Current consumption	approx. 140 has
Supply voltage:	USB: 5 V Non-adjusted input: 6 V to 10 V
Communication interfaces	UART /SPI /I2C/USB/Digital/ Analog
Integrated light source	approx. 20 lumens
Dimensions and Weight	42 x 38 x 15 mm / 10 g

Table 1 PixyCam ver. 2 CMUcam5 Specification

3.2.1.2 Arduino uno

We are using Arduino Uno R3 (**Figure 3.1**), which is one of the Arduino boards with the features in **Table 2** [35]:

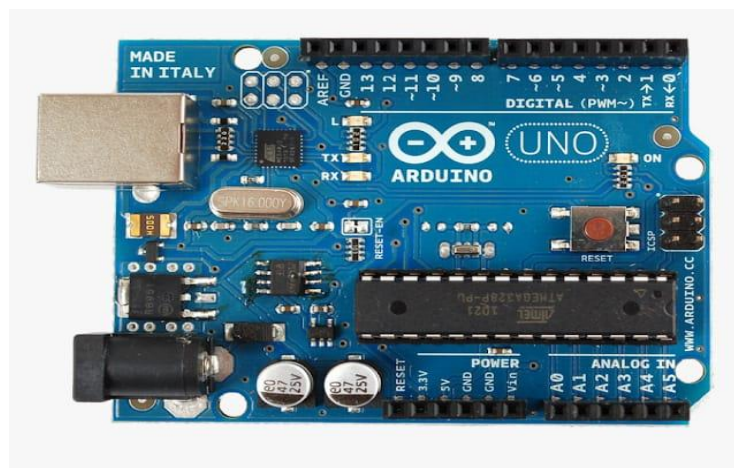


Figure 3. 1 Arduino R3

Microcontroller	ATmega328P
Operating Voltage	5V
Input Voltage (recommended)	7-12V
Input Voltage (limit)	6-20V
Digital I/O Pins	14 (of which 6 provide PWM output)
PWM Digital I/O Pins	6
Analog Input Pins	6
DC Current per I/O Pin	20 mA
DC Current for 3.3V Pin	50 mA
Flash Memory	32 KB (ATmega328P) of which 0.5 KB used by bootloader
SRAM	2 KB (ATmega328P)
EEPROM	1 KB (ATmega328P)
Clock Speed	16 MHz
LED_BUILTIN	13
Length	68.6 mm
Width	53.4 mm
Weight	25

Table 2 Arduino Uno R3 Features

3.2.1.3 Motor Driver L298N

It is an electronic circuit (**Figure 3.2**) for controlling the speed and direction of rotation of two motors simultaneously. This circuit can handle a current of up to 2A and a voltage ranging from 5 to 35 volts [36].

The Enable A and Enable B ports allow for adjusting the voltage supplied to the motors using a PWM signal and can be connected to a 5V random mode when speed control is not needed.

The Input 1 and Input 2 ports for motor A and Input 3 and Input 4 for motor B enable control of the H-bridge and thus the direction of rotation of the motors (**Figure 3.3**) [36]

A jumper is connected to power the 5V output in the circuit.

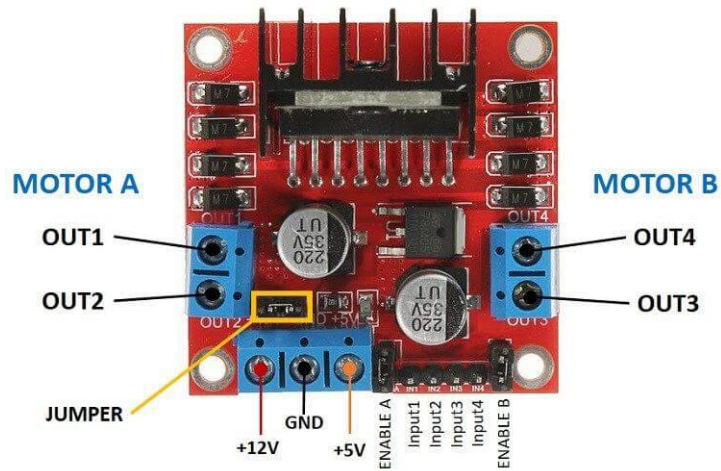


Figure 3. 2 Motor Driver L298N

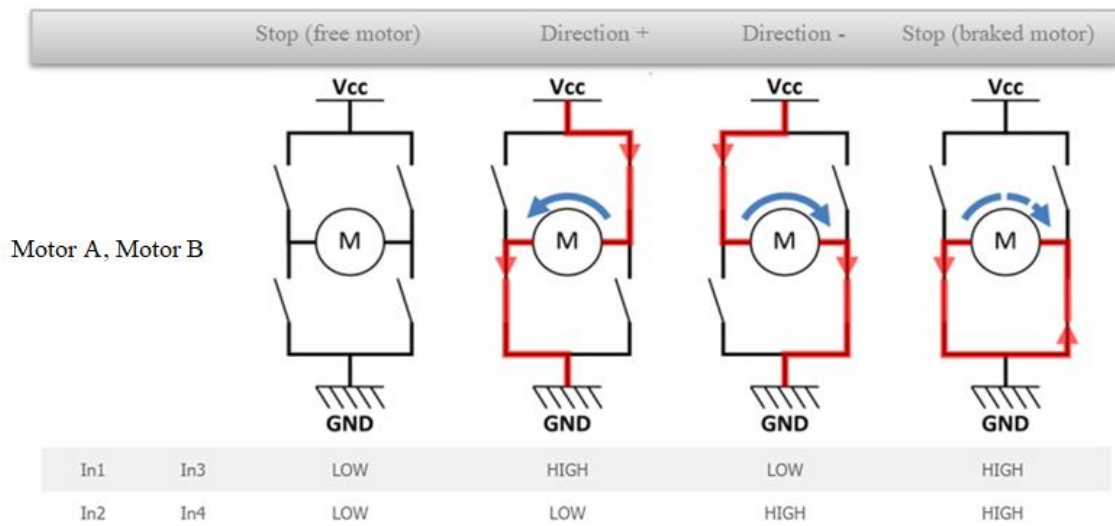


Figure 3. 3 Operating principle of the H-bridge

3.2.1.4 DC Motors

We used two TT DC Gearbox Motors (**Figure 3.4**). The characteristics of these motors [37] are as showing in **Table 3**

Size	70 * 22.5 * 19mm (excluding output shaft, one side output shaft: 8.5 mm)
Reduction ratio	1:48
Operating voltage	3V - 6V
On 3v voltage	current 70ma speed about 95 r / min
On 6v voltage	current 80ma speed 200 r / min
Unload current	≤200mA on 6V, ≤150mA on 3V
Unload speed	200 ± 10 RPM at 6V, 90 ± 10 RPM at 3V

Table 3 TT DC Gearbox Motors characteristics

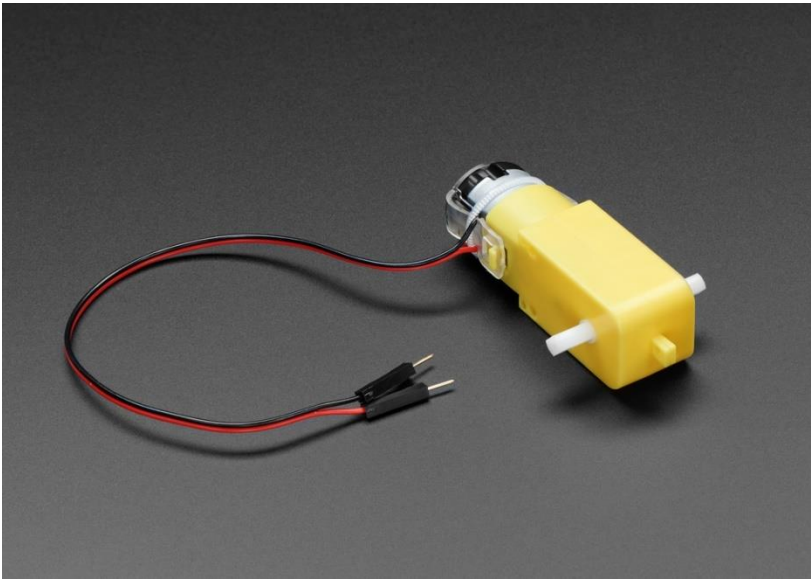


Figure 3. 4 TT DC Gearbox Motors

3.2.2 Circuit

The electrical circuit of the robot with the PixyCam is shown in **(Figure 3.5)**

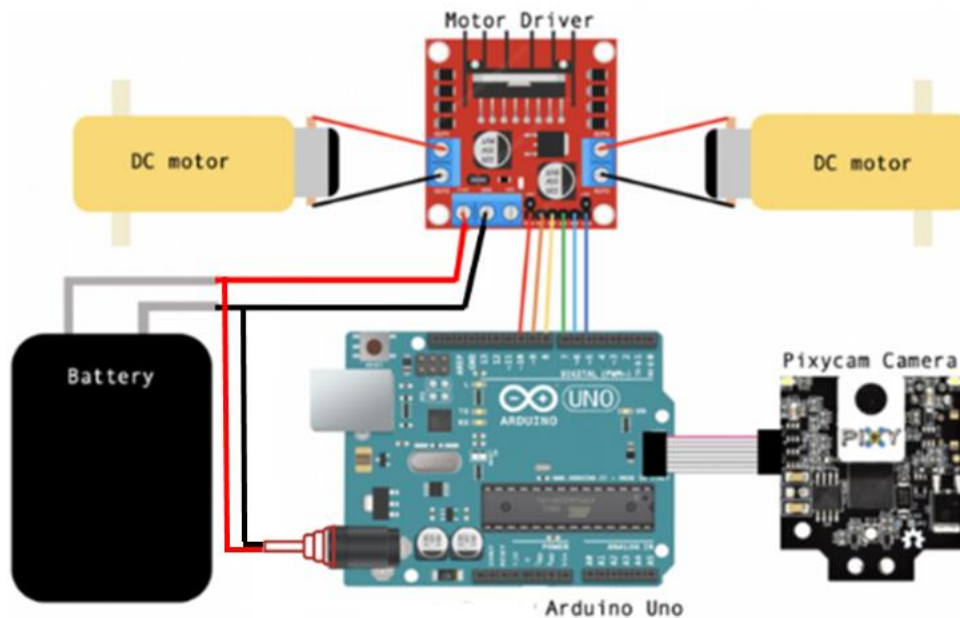


Figure 3. 5 The Robot circuit using Pixycam

3.2.3 Program and Speed Control

We used the Arduino IDE environment to program an Arduino board for tracking. This environment provides ready-made libraries for cameras such as Pixycam or ESP32, along with easy-to-follow instructions. The program operates based on the position of the Scorpion in real-time camera images and moves towards it. The program relies on only one axis, the x-axis.

3.2.3.1 Speed without control

To simplify motion control. Initially, we started with a program that did not include speed control but relied on a simple logic for turning and moving forward, operating according to the Flowchart showing in **(Figure 3.6)**

Where the robot's turning or moving forward is done at the maximum speed (200)

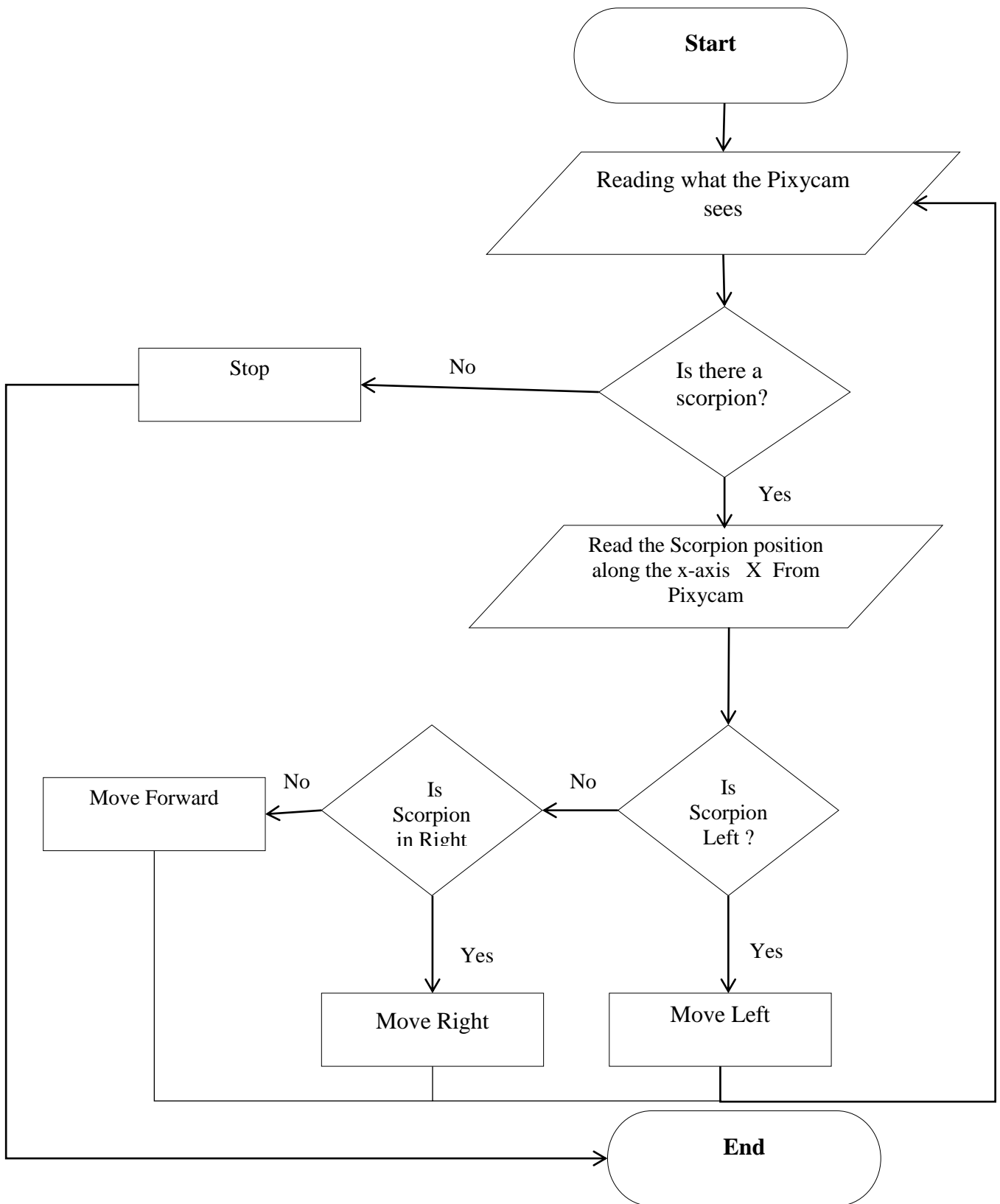


Figure 3. 6 Flowchart of the robot without control

The real problem with this program is that the robot's tracking of the scorpion is not smooth and moves in a step-like manner. Turning at the maximum speed causes the robot to turn at a larger angle than necessary, making the target appear on the opposite side. This results in the robot turning from the other side in the same way, leading to the same problem. Consequently, the robot keeps turning right and left in an attempt to track.

So we attempted to make the robot's movement smoother by controlling the turning speed.

3.2.3.2 PID Speed Control

To control the turning speed, we used a PID controller (**Figure 3.7**).

- Proportional (P): Increases system response speed based on current error.
- Integral (I): Significantly reduces steady-state error, thereby minimizing final error and increasing system accuracy.
- Derivative (D): Reduces sudden errors and oscillations.

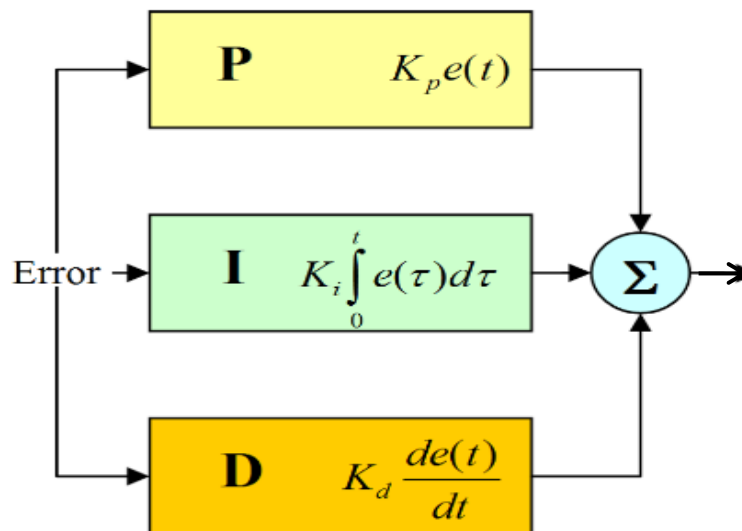


Figure 3. 7 The PID controller

The basic principle of the program with this controller is to calculate the error between the center of the image (160) and the position of the scorpion in the image. If the scorpion is not in the center, we determine the direction of the turn. Then, based on the error margin (whether the scorpion is far from or close to the center), we adjust the turning speed, ensuring smooth rotation of the robot. The program operates according to the Flowchart in (**Figure 3.8**)

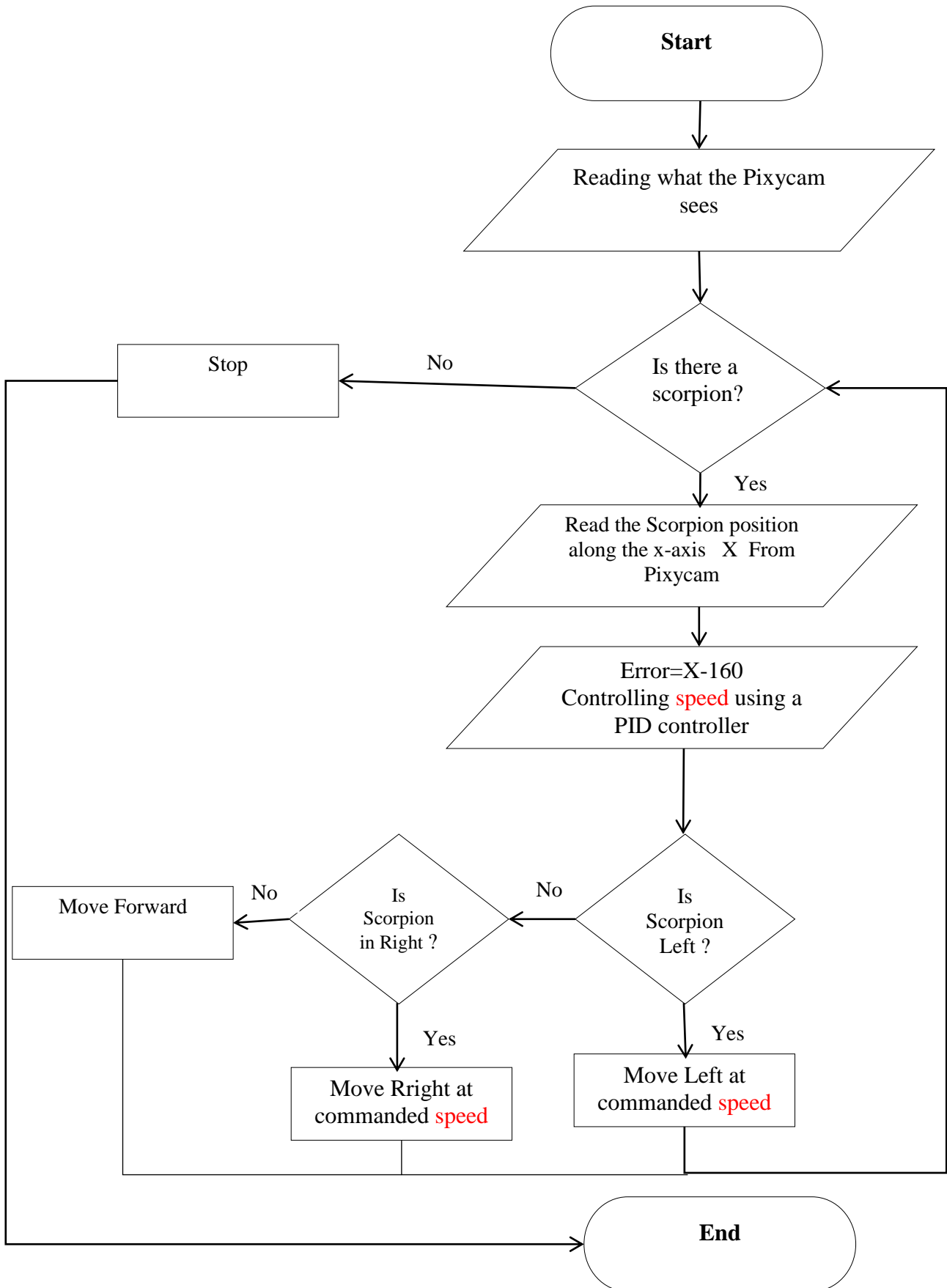


Figure 3. 8 Flowchart of the robot With PID control

In this controller, we selected the parameters K_p , K_i , and K_d through trial and error until we reached suitable parameters and satisfactory results (**Tabel 4**). After using the program, we indeed achieved good performance, smooth rotation, and excellent tracking. The downside of this controller is that the parameters are not ideal since we did not calculate them theoretically. Because the Arduino and the circuit used are somewhat complex and require time to calculate the optimal parameters, we transitioned to using a controller that does not require knowledge of the circuit characteristics and treats the system as a black box, which is fuzzy logic control.

K_p	0.2
K_i	0.01
K_d	0.1

Table 4 PID controller gains

3.2.3.3 Fuzzy Logic Speed Control

In this final stage of program refinement, we utilized fuzzy logic to adjust the rotation speed. Fuzzy logic (**Figure 3.9**) is a form of multi-valued logic where the truth value of any variable can be a real number between 0 and 1. Fuzzy logic was designed by the Iranian-American scientist Lotfi A. Zadeh. He conceived and introduced the concept of fuzzy logic for the first time in 1965 through his famous paper titled "Fuzzy Sets," published in the journal "Information and Control." [41] It is used to process imprecise or uncertain information and aims to simulate human reasoning in dealing with ambiguous or unclear data .

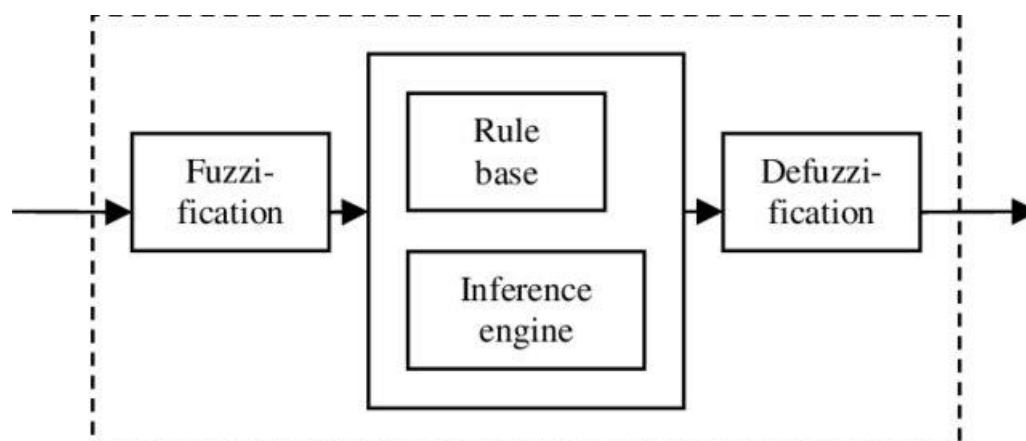


Figure 3. 9 General structure of a fuzzy controller

The fuzzy logic control program for the robot has a flowchart similar to the PID control program, except for the method used to calculate the commanded speed

The inputs to the fuzzy system were the error and its derivative, and the outputs were rotational speeds: slow, medium, and fast.

We performed fuzzification of the inputs (error and the error derivative)(**Figure 3.10**) and output (Speed) (**Figure 3.11**) into three fuzzy subsets with trapezoidal types of membership functions as follows

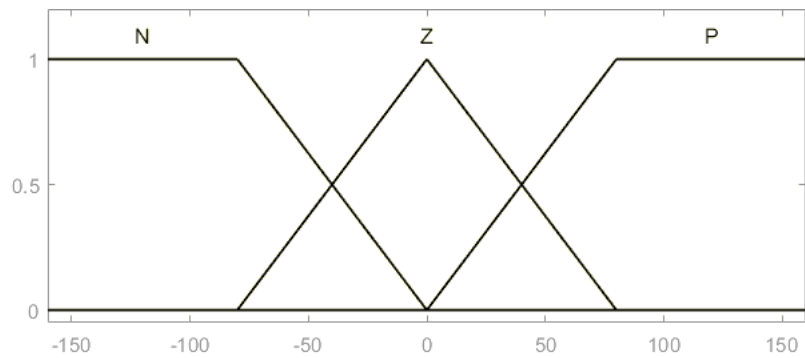


Figure 3. 10 Error and the Error Derivative Fuzzification

N : Negative

Z: Zero

P:Positive

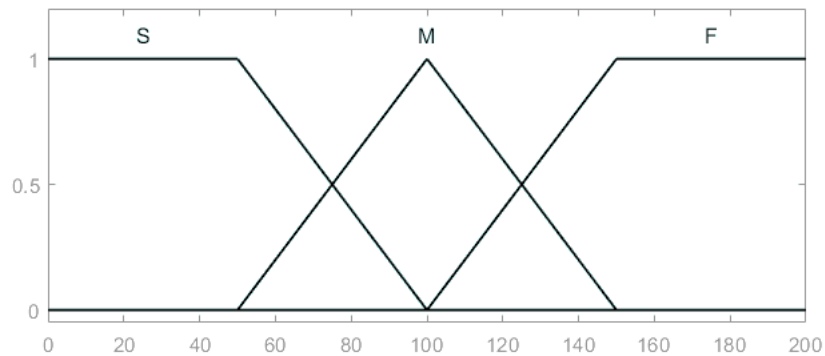


Figure 3. 11 Speed Fuzzification

S : Slow

M : Mediem

F :Fast

The inference table is given in **Table 5** :

Speed		de		
		N	Z	P
e	N	M	M	F
	Z	S	S	M
	P	M	M	F

Table 5 Inference Table

In Arduino, defuzzification is performed using the function `defuzzify()`, and the library employs the Center of Gravity method for defuzzification of variables [40] .

Fuzzy logic control provides us with good precision in operation and optimal tracking.

3.3 The second method :

In this case, we used an ESP32 camera, which, unlike the PixyCam, is not pre-trained. We trained it using artificial intelligence and then connected it to the circuit to achieve higher accuracy in tracking.

3.3.1 Components

In this method, we used the ESP32-Cam With Arduino Uno And the same motors and Motor driver L298N

We used **ESP32 AI Thinker Model** with its corresponding development board to upload the program, then disconnected it and connected it to the Motor Driver L298N. This camera is characterized by the following features [38] :

- Ultra-small 802.11b/g/n Wi-Fi + BT/BLE SoC module
- Low-power dual-core 32-bit CPU for application processors
- Up to 240MHz, up to 600 DMIPS
- Built-in 520 KB SRAM, external 4M PSRAM
- Supports interfaces such as UART/SPI/I2C/PWM/ADC/DAC
- Support OV2640 and OV7670 cameras with built-in flash
- Support for images WiFi upload
- Support TF card
- Support multiple sleep modes
- Embedded Lwip and FreeRTOS
- Support STA/AP/STA+AP working mode
- Support Smart Config/AirKiss One-click distribution network
- Support for serial local upgrade and remote firmware upgrade (FOTA)

3.3.2 Circuit

The electrical circuit of the robot with the ESP32-CAM is illustrated in (Figure 3.12)

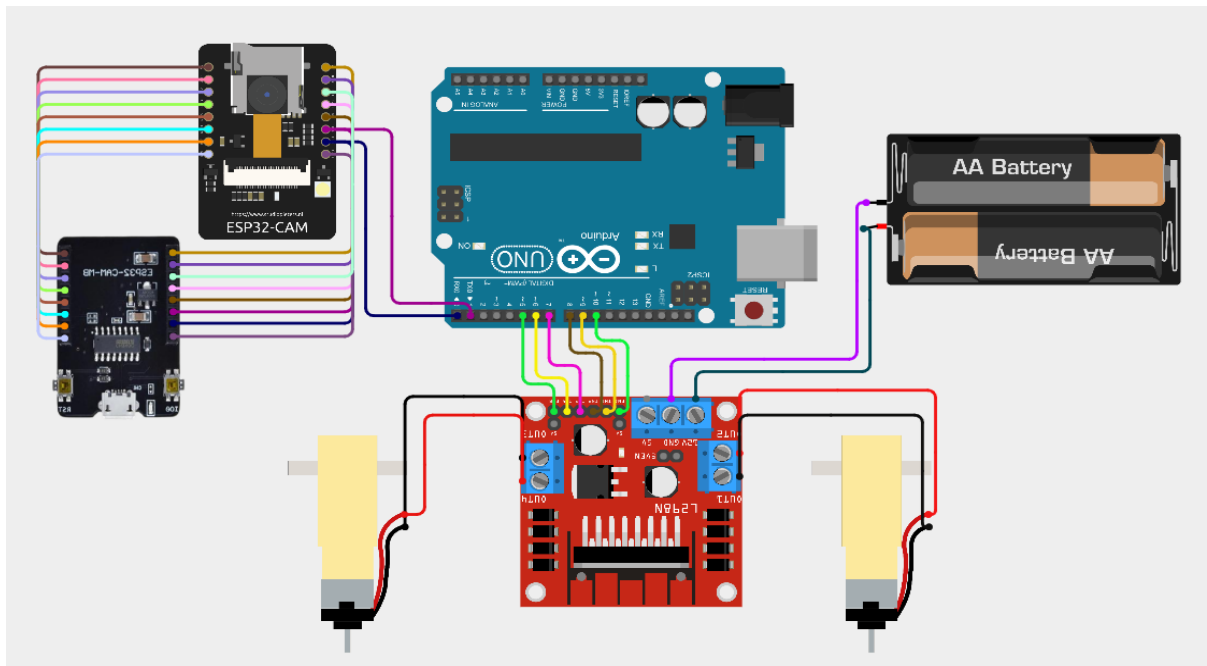


Figure 3. 12 The Robot circuit using ESP32-Cam

3.3.3 ESP32CAM Training

We created a neural network to train the camera and then deployed it as an Arduino Library using Edge Impulse , which is an AI platform established in 2019 by Zach Shelby and Jan Jongboom, offering comprehensive capabilities for: collecting data from various sources, analyzing data, training machine learning models, and deploying models on different devices such as microcontrollers and FPGAs [39].

To train the model, we followed these steps

3.3.3.1 Collection of Data

After creating an account on Edge Impulse, the first step is data collection, and this is done by capturing a set of images of the object, in our case the scorpion, at various angles and positions, using two methods:

- using the computer's camera, or alternatively, by using a phone after scanning a QR code.
- involves uploading existing data from the computer.

We have opted to use the first method and we captured 47 images of the scorpion in different positions using the phone

3.3.3.2 Data Cleaning

In this step, we clean the data, review the captured images, and remove any unclear images.

Then we split the data into 81% (38 images) for training and 19% (9 images) for testing (Figure 3.13).

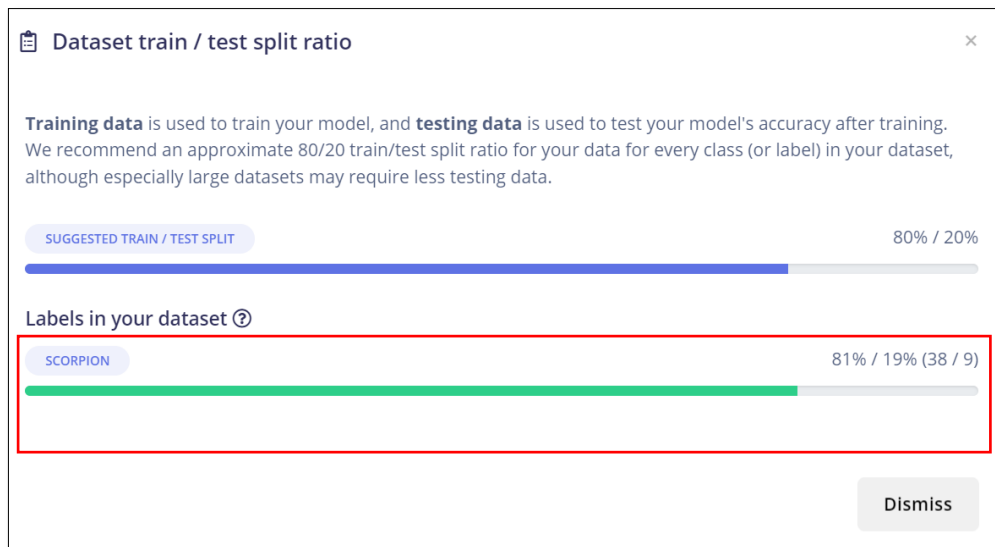


Figure 3. 13 Training and Testing data

we label the dataset and accurately identify the scorpion in each image (Figure 3.14):



Figure 3. 14 Scorpions identification

3.3.3.3 Model building

We start by initiating the initial model construction by selecting its key features by selecting the type of input data, which in our case is scorpion images which we have already uploaded, it's also possible to adjust the size of these images to reduce the size of the model, and classifying the output data (whether a scorpion is present or not). We also need a processing block and a learning block (**Figure 3.15**).

In Edge Impulse, there is one available block in the processing block (**Image**) when it comes to image processing, and it is a block that preprocesses and normalizes image data, working on enhancing colors to increase neuron performance accuracy.

For the learning block, we opted for the first option, which is Object Detection (Images).

This block works accurately for object detection models and ensures good performance even with relatively small image datasets. Moreover, it is compatible with all boards unlike the other available block, which only works with the BrainChip AKD1000 MINI PCIe board [39].

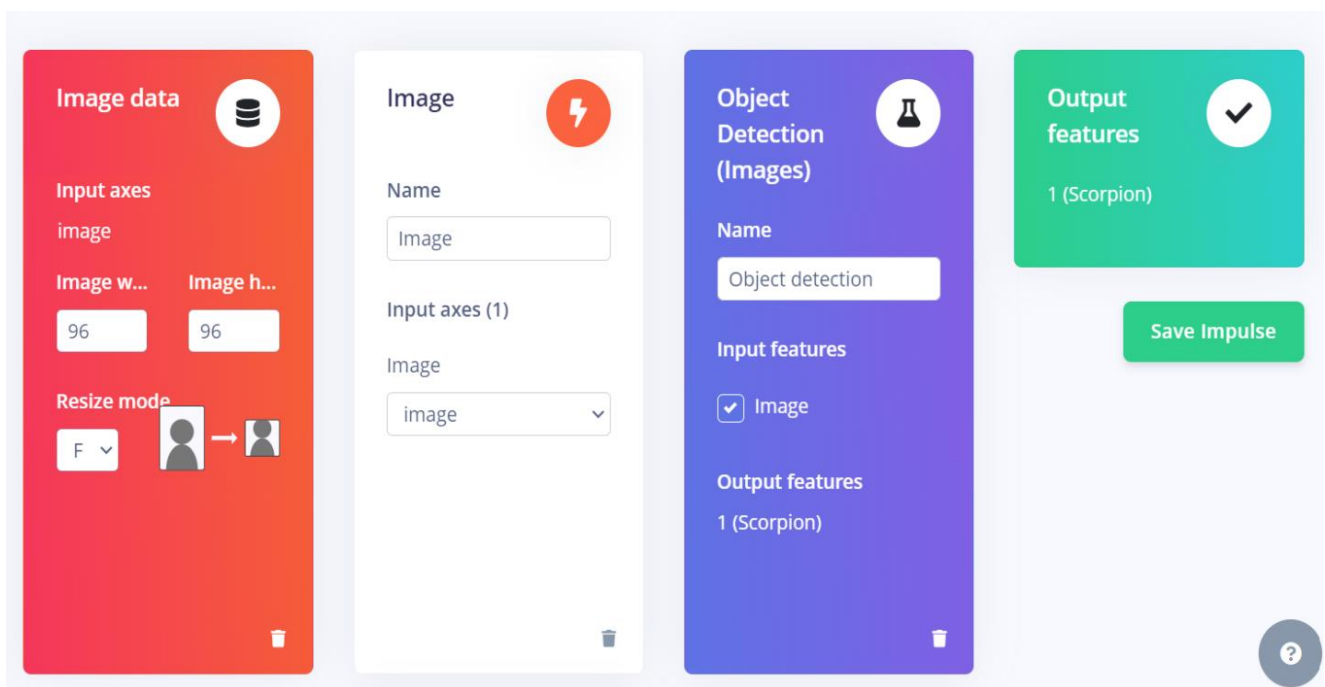


Figure 3. 15 Initial model construction

After saving the previous options, we generate the features , and in the space **(Figure3.16)** all the scorpions (all the blue points) appear clustered together, indicating that similar features have been captured from the images .

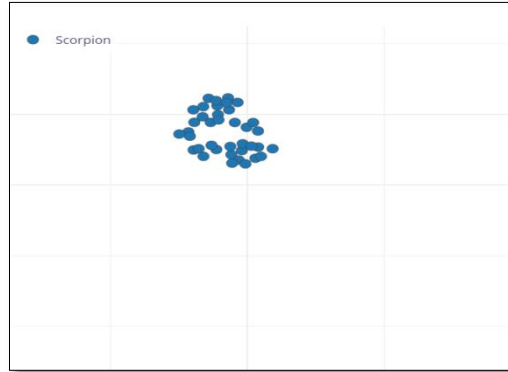


Figure 3. 16 Feature Explorer

Finally and After the initial construction and feature Generation , We configure the settings for the Neural Network, and then we begin the training.

We trained the neuron with the settings showing in **(Figure 3.17)** :

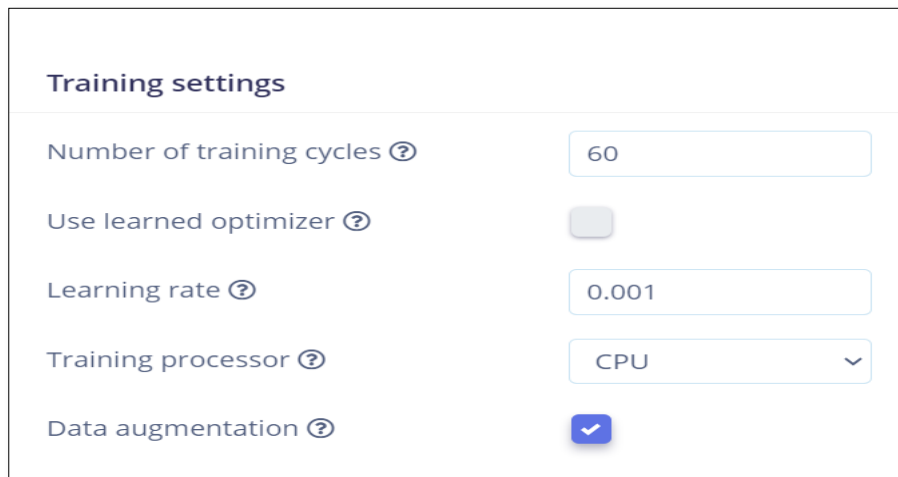
A screenshot of a 'Training settings' panel. It contains five rows of settings: 'Number of training cycles' with a value of 60; 'Use learned optimizer' with a disabled toggle switch; 'Learning rate' with a value of 0.001; 'Training processor' with a dropdown menu set to 'CPU'; and 'Data augmentation' with a checked checkbox. Each setting has a help icon (a question mark in a circle) to its right.

Figure 3. 17 Neural Network Settings

Regarding the Neural Network architecture, we chose FOMO (Faster Objects, More Objects) MobileNetV2 0.1, which is a small-sized model (less than 100 KB) and is more accurate compared to other small-sized models [39].

3.3.3.4 Model Evaluation

After training the model, it is evaluated. This is what we obtained after evaluating our model (**Figure 3.18**) :

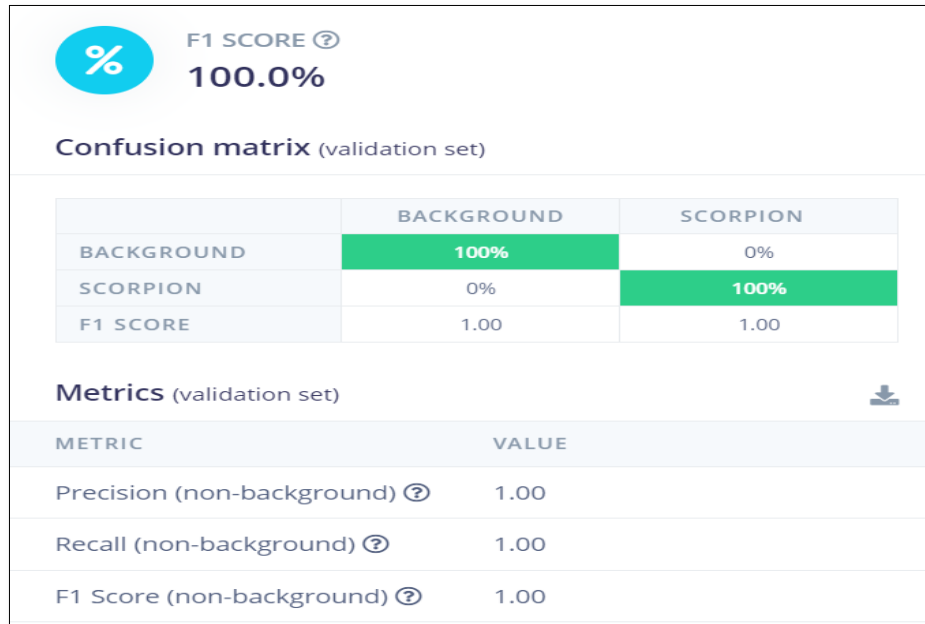


Figure 3. 18 Model Evaluation

3.3.3.5 Model Deployment

At the end of the process, to make the model usable in an Arduino program, we deploy the model as an Arduino library, a feature provided by Edge Impulse. This ensures the model has the characteristics are shown in **Table 6** :

	IMAGE	OBJECT DETECTION	TOTAL
LATENCY	15 ms.	1 000 ms.	1 015 ms.
RAM	4,0K	235,5K	235,5K
FLASH	-	64,3K	-
ACCURACY			-

Table 6 Model characteristics

3.3.4 Program

In this program, the general principle is similar to the program using the Pixycam. The fundamental difference is that in this case, the camera is programmed first and then connected to the Arduino. Therefore, there are two programs: one for the camera and one for the Arduino. The camera program primarily relies on the CNN model we trained, whereas the Arduino program is no different from the one used with the Pixycam. In both cases, the object's position along the x-axis is determined by the camera, which then dictates the speed of rotation and movement directions.

3.4 Shape and external design

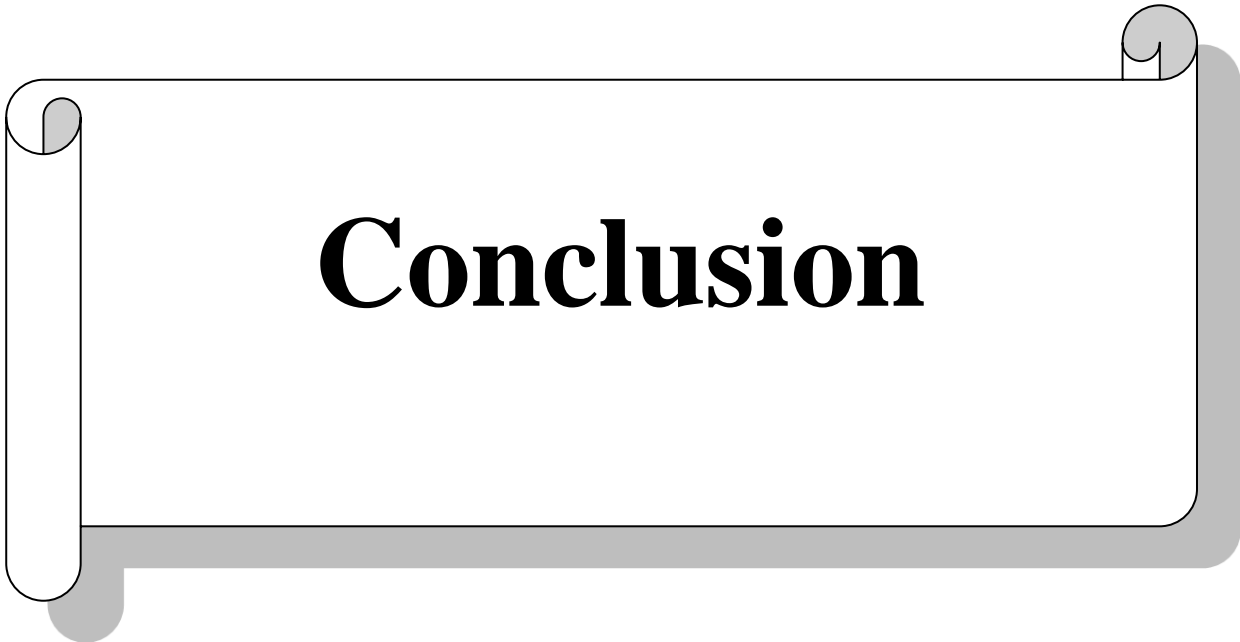
We aimed to design an external shape as a rubber container that can store the poison. However, due to the lack of resources, this is the approximate shape we are striving to achieve, but in rubber (**Figure 3.19**).



Figure 3. 19 The Robot external design

3.5 Conclusion

In this chapter, we identified the hardware and software components of the robot, detailed the characteristics of each component, explained the robot programming method, control using PID correction and fuzzy logic, and also trained the ESP32-CAM using CNN.



Conclusion

This thesis introduces a state-of-the-art technical solution for the safe extraction of venoms from snakes and scorpions, leveraging advanced robotics technology. The proposed robotic system, engineered with Arduino and outfitted with two distinct types of cameras, operates autonomously within snake and scorpion farms.

Key features of the robot include:

- **Autonomous Navigation:** Utilizing Arduino-based commands, sophisticated motion controllers, and cutting-edge artificial intelligence (AI) techniques, the robot adeptly maneuvers through the farm environment.
- **Scorpion Detection:** The system employs Convolutional Neural Networks (CNN) for accurate scorpion detection, facilitated by the ESP32 camera. This AI integration significantly enhances the precision of the camera, ensuring reliable identification and tracking of scorpions.
- **Optimized Movement:** Speed controllers are implemented to maintain smooth and efficient robot motion, which not only improves operational stability but also minimizes energy consumption.

By integrating these advanced technologies, the robotic system provides a modern and efficient method for venom extraction, improving both safety and efficacy. Additionally, the innovative design and functionality of this project have been recognized, as it has been officially registered with the Algerian National Institute of Industrial Property (INAPI) for patent issuance.

Overall, this thesis demonstrates a significant advancement in the field of robotics and venom extraction, showcasing the potential of AI and autonomous systems in enhancing agricultural and biomedical practices.

Future work includes enhancing AI for better species detection and real-time processing, developing precise robotic handling and automated sample collection, and innovative design improvements for greater efficiency. Additionally, integrating environmental sensors and exploring renewable energy sources will enhance sustainability and adaptability.

References

- [1]. world health organization .<https://www.who.int/news-room/fact-sheets/detail/snakebite-envenoming>
- [2]. J. P. Chippaux, M. Goyffon, (2008), Epidemiology of scorpionism: A global appraisal, *Acta Trop*, 107(2), 71-79
- [3]. Sadine Salah Eddine , Djilani Salma , Kerboua Kheir Eddine , (2020), Overview on Scorpions of Algeria : *ALGERIAN JOURNAL OF HEALTH SCIENCES* ,
- [4]. Bardy.A, Younes. M, Sarhan. M.M.H , Saleh. M, (2018). On the scorpion fauna of Egypt, with an identification key (Arachnida: Scorpiones). *Zoology in the Middle East*, 64, 75-87
- [5]. [Online] <https://worldpopulationreview.com/country-rankings/snake-population-by-country>
- [6]. [Online] <https://www.animalia.bio/egyptian-cobra>
- [7]. [Online] <https://www.hummingbirdsplus.org/nature-blog-network/3-types-of-venomous-snakes-in-algeria/>
- [8].Sharp, Jay. "The Desert Horned Viper (Cerastes cerastes)." DesertUSA, www.desertusa.com/animals/horned_viper.html
- [9].Schehrazad Selmane, Mohamed Lhadj , (2022) , Scorpion Envenomations in Algeria, *Journal of Preventive, Diagnostic and Treatment Strategies in Medicine*
- [10]. world population review , <https://worldpopulationreview.com/country-rankings/snake-bite-deaths-by-country>.
- [11].Chippaux, J.P. (2009). Global incidence and management of snake and scorpion envenomations. *Médecine/Sciences*
- [12].Tadeusz Plusa , Katarzyna Smędzik,(2015), The threat of snake and scorpion venoms , National library of medicine
- [13].Ahmadi, S., Knerr, J. M., Argemi, L., Bordon, K. C. F., Pucca, M. B., Cerni, F. A., Arantes, E. C., Çalışkan, F., & Laustsen, A. H. (2020). Scorpion Venom: Detriments and Benefits. *Biomedicines*
- [14].Pal, S.K., Gomes, A., Dasgupta, S.C., & Gomes, A. (2002). Snake venom as therapeutic agents: From toxin to drug development. *Indian Journal of Experimental Biology*
- [15].N.Oukkache, F. Chgoury, M. Lalaoui, A. Alagón Cano , N. Ghalim ,(2013),"Comparison between two methods of scorpion venom milking in Morocco. *Journal of Venomous Animals and Toxins including Tropical Diseases*.

- [16]. [Online] <https://www.creative-proteomics.com/enom/about>
- [17]. Emanuel Brenes, Aarón Gómez, 2016, Scorpion maintenance in captivity for venom extraction in Costa Rica , Revista de Biología Tropical" (Rev Biol Trop)
- [18]. Berger, Raymond D. 1975. "Snake Venom Milker." United States Patent 3,926,149.
- [19]. Mouad MKAMEL , Rachid Saile , Omar Tanane , and Anass Kettani, 2022, The robotic scorpion venom extraction system, Revue de 'Entrepreneuriat et de l'Innovatio
- [20]. P. Rajeshwari, P. Abhishek, P. Srikanth, T. Vinod, 2019, Object Detection: An Overview, International Journal of Trend in Scientific Research and Development (IJTSRD) .
- [21]. [Online] , Introduction to object detection with deep learning, 2023
<https://www.superannotate.com/blog/object-detection-with-deep-learning>
- [22]. [Online] Artem Oppermann, Artificial Intelligence vs. Machine Learning vs. Deep Learning: What's the Difference? , 2023, <https://builtin.com/artificial-intelligence/ai-vs-machine-learning>
- [23]. *KERBOUCHE Daoud* . (2023). (Master's thesis). Department of Automatics and Electromechanics , University Of Ghardaia
- [24]. [Online] . <https://www.kubii.com/en/content/72-what-is-raspberry-pi>
- [25]. [Online]. <https://www.projectcompany.org/ar/product-page/raspberry-pi-4-b-128g-sd-case-pi4-2g-ram-ddr4-4-core-cpu-1-5-ghz>
- [26]. [Online] pixy Documentation.
<https://docs.pixycam.com/wiki/doku.php?id=wiki:v1:overview>
- [27]. [Online]. ESP32 Cam : PinOut, Specifications, Types, Interfacing & Its Applications .
<https://www.elprocus.com/esp32-cam/>
- [28]. [Online]. <https://ifuturetech.org/product/esp32-cam-mb-micro-usb-to-serial-converter-loader/>
- [29]. [Online]. <https://hobbypcb.com/products/ftdi-usb-to-ttl-serial-adapter-3-3v-and-5v>
- [30]. [Online]. Shanthababu Pandian, 2020, Machine Learning Process_Overview,
<https://medium.com/analytics-vidhya/machine-learning-process-overview-1daa05c30150>

[31]. Yi Li, Min Liu , Dandan Jiang ,2022, Application of Unmanned Aerial Vehicles in Logistics: A Literature Review , *Sustainability*

[32].[Online]. <https://www.fieldrobotics.it/>

[33].[Online].<https://emergencydroneresponder.com/10-impressive-examples-of-rescue-robots-you-should-be-aware-of/>

[34].[online]. Camera PixyCam ver. 2 CMUcam5 - a smart image sensor - SparkFun SEN-14678. [**Camera PixyCam ver. 2 CMUcam5 - a smart image Botland - Robotic Shop**](#)

[35].[Online]. Arduino official website . https://store.arduino.cc/products/arduino-uno-rev3?_gl=1*1ng7ojk*_gcl_au*MTc4MjE2MTQ2MS4xNzE2OTkwOTA2*FPAU*MTc4MjE2MTQ2MS4xNzE2OTkwOTA2*_ga*MTAyMTMwMzE1MS4xNjk3NTM4Mjk2*_ga_NEXN8H46L5*MTcxNzg0NjQ2Mi4xMy4xLjE3MTc4NDczMjAuMC4wLjIxODg4MDU0NA..*_fplc*aWZUJTJCS3BSWmFZVUFhQmRTN1RvSXQyR3RWczhjS294ZXFvZ290Wk9HclYlMkYweWJyRDBLUCUyQjRaZk9UM1JkMmpXR25HTjVnME5GcjNtc3VOY0NtbzBYRFE5OGNyWUVSaG03Q0s5NV11d0FXelEyWXVwNW9tTE5Mb24xZFdCRSUyQlEIM0QlM0Q.

[36]. [Online]. CFAURY. Pont en H L298N . <https://arduino.blaisepascal.fr/pont-en-h-l298n/>

[37].[Online]. <https://www.hatfon.com/urun/tek-eksen-disli-reduktor-motor-dc-3v-6v>

[38].[Online].<https://robu.in/product/ai-thinker-esp32-cam-development-board-wifiblueetooth-with-ov2640-camera-module/>

[39].[Online].Edge Impulse Official website. <https://edgeimpulse.com/>

[40].[Online]. <https://blog.zerokol.com/2012/09/arduinofuzzy-fuzzy-library-for-arduino.html>

[41]. L.A._Zadeh ,1965, Fuzzy Sets, journal of Information and Control

Appendix A – Source Code(Pixycam)

```
#include <Pixy2.h>
// Define PixyCam
Pixy2 pixy;
// Define motor pins in the L298N module
const int ENA = 5; // Speed pin for motor 1
const int IN1 = 7;
const int IN2 = 6;
const int ENB = 10; // Speed pin for motor 2
const int IN3 = 9;
const int IN4 = 8;

// Maximum movement speed
const int MAX_SPEED = 200;
// Width of the central area
const int CENTER_WIDTH = 40;

// Motor setup
void setupMotors() {
    pinMode(ENA, OUTPUT);
    pinMode(IN1, OUTPUT);
    pinMode(IN2, OUTPUT);
    pinMode(ENB, OUTPUT);
    pinMode(IN3, OUTPUT);
    pinMode(IN4, OUTPUT);
}

// Setup function
void setup() {
    Serial.begin(115200);
    pixy.init();
}
```

```
        setupMotors();
    }

void moveForward() {
    Serial.println("Forward.");
    digitalWrite(IN1, HIGH);
    digitalWrite(IN3, LOW);
    analogWrite(ENA, MAX_SPEED);
    digitalWrite(IN2, LOW);
    digitalWrite(IN4, HIGH);
    analogWrite(ENB, MAX_SPEED);
}

void moveLeft() {
    Serial.println("Left.");
    // Change setting to rotate left
    digitalWrite(IN1, LOW);
    digitalWrite(IN3, LOW);
    analogWrite(ENA, MAX_SPEED);
    digitalWrite(IN2, HIGH);
    digitalWrite(IN4, HIGH);
    analogWrite(ENB, MAX_SPEED);
}

void moveRight() {
    Serial.println("Right.");
    // Change setting to rotate right
    digitalWrite(IN1, HIGH);
    digitalWrite(IN3, HIGH);
    analogWrite(ENA, MAX_SPEED);
    digitalWrite(IN2, LOW);
    digitalWrite(IN4, LOW);
}
```

```

    analogWrite(ENB, MAX_SPEED);
}

void Stop() {
    Serial.println("Stop.");
    digitalWrite(IN1, LOW);
    digitalWrite(IN3, LOW);
    analogWrite(ENA, 0);
    digitalWrite(IN2, LOW);
    digitalWrite(IN4, LOW);
    analogWrite(ENB, 0);
}

// Main loop function
void loop() {
    pixy.ccc.getBlocks();

    if (pixy.ccc.numBlocks) {
        int x = pixy.ccc.blocks[0].m_x;

        // Display x value in Serial Monitor
        Serial.print("Object X Position: ");
        Serial.println(x);

        // Control the robot's direction based on the target's position
        if (x < 140 ) { // Target is to the left
            moveLeft();
        } else if (x > 180 ) { // Target is to the right
            moveRight();
        } else { // Target is in the center
            moveForward();
        }
    }
}

```

```
    } else {  
        // If no target is found, stop the robot  
        Stop();  
    }  
}
```

Appendix B – Source Code(PID-Pixycam)

```
#include <Pixy2.h>

// Define PixyCam

Pixy2 pixy;

// Define motor pins in the L298N module

const int ENA = 5;

const int IN1 = 7;

const int IN2 = 6;

const int ENB = 10;

const int IN3 = 9;

const int IN4 = 8;

// Maximum movement speed and width of the central area

const int MAX_SPEED = 200;

const int CENTER_WIDTH = 40;

int speed;

// PID variables

float Kp = 0.2; // Proportional constant

float Ki = 0.01; // Integral constant

float Kd = 0.1; // Derivative constant

int previous_error = 0;

float integral = 0;
```

```
// Motor setup

void setupMotors() {

    pinMode(ENA, OUTPUT);

    pinMode(IN1, OUTPUT);

    pinMode(IN2, OUTPUT);

    pinMode(ENB, OUTPUT);

    pinMode(IN3, OUTPUT);

    pinMode(IN4, OUTPUT);

}

// Setup function

void setup() {

    Serial.begin(115200);

    pixy.init();

    setupMotors();

}

void moveForward(int speed) {

    Serial.println("Forward.");

    digitalWrite(IN1, HIGH);

    digitalWrite(IN3, LOW);

    analogWrite(ENA, speed);

    digitalWrite(IN2, LOW);

    digitalWrite(IN4, HIGH);

}
```

```
    analogWrite(ENB, speed);  
}
```

```
void moveLeft(int speed) {  
    Serial.println("Left.");  
    digitalWrite(IN1, LOW);  
    digitalWrite(IN3, LOW);  
    analogWrite(ENA, speed);  
    digitalWrite(IN2, HIGH);  
    digitalWrite(IN4, HIGH);  
    analogWrite(ENB, speed);  
}
```

```
void moveRight(int speed) {  
    Serial.println("Right.");  
    digitalWrite(IN1, HIGH);  
    digitalWrite(IN3, HIGH);  
    analogWrite(ENA, speed);  
    digitalWrite(IN2, LOW);  
    digitalWrite(IN4, LOW);  
    analogWrite(ENB, speed);  
}
```

```
void Stop() {
```

```

Serial.println("Stop.");

digitalWrite(IN1, LOW);

digitalWrite(IN3, LOW);

analogWrite(ENA, 0);

digitalWrite(IN2, LOW);

digitalWrite(IN4, LOW);

analogWrite(ENB, 0);

}

// Main loop function

void loop() {

  pixy.ccc.getBlocks();

  if (pixy.ccc.numBlocks) {

    int x = pixy.ccc.blocks[0].m_x;

    int error = x - 160; // 160 is the center of the image

    integral += error;

    int derivative = error - previous_error;

    int speed = Kp * error + Ki * integral + Kd * derivative; // PID calculation

    speed = constrain(speed, 0, MAX_SPEED); // Ensure the speed is within allowed
limits

    // Display x value in Serial Monitor

    Serial.print("Object X Position: ");

    Serial.println(x);

```



```
// Control the robot's direction based on the target's position

if (error < -CENTER_WIDTH) {
    moveLeft(speed);
} else if (error > CENTER_WIDTH) {
    moveRight(speed);
} else {
    moveForward(MAX_SPEED);
}

previous_error = error; // Update previous error for the next iteration
} else {
    // If no target is found, stop the robot
    Stop();
}
}
```

Appendix C – Source Code (Fuzzy Logic-Pixycam)

```
#include <Pixy2.h>

#include <Fuzzy.h>

// Define PixyCam

Pixy2 pixy;

// Define motor pins in the L298N module

const int ENA = 5;

const int IN1 = 7;

const int IN2 = 6;

const int ENB = 10;

const int IN3 = 9;

const int IN4 = 8;

// Maximum movement speed and width of the central area

const int MAX_SPEED = 200;

const int CENTER_WIDTH = 40;

// Tracking variables

unsigned long lastUpdateTime = 0;

int previous_error = 0;

// Motor setup

void setupMotors() {

    pinMode(ENA, OUTPUT);
```

```

pinMode(IN1, OUTPUT);

pinMode(IN2, OUTPUT);

pinMode(ENB, OUTPUT);

pinMode(IN3, OUTPUT);

pinMode(IN4, OUTPUT);

}

// Fuzzy logic setup

Fuzzy *fuzzy = new Fuzzy();

void setupFuzzyLogic() {

    // Input variable: error

    FuzzyInput *error = new FuzzyInput(1);

    FuzzySet *negative = new FuzzySet(-160, -160, -80, 0); // Trapezoidal

    FuzzySet *zeroError = new FuzzySet(-10, 0, 0, 10); // Triangular

    FuzzySet *positive = new FuzzySet(0, 80, 160, 160); // Trapezoidal

    error->addFuzzySet(negative);

    error->addFuzzySet(zeroError);

    error->addFuzzySet(positive);

    fuzzy->addFuzzyInput(error);

    // Input variable: derivative of error

    FuzzyInput *derivative = new FuzzyInput(2);

    FuzzySet *derivativeNegative = new FuzzySet(-160, -160, -80, 0); // Trapezoidal

```

```

FuzzySet *derivativeZero = new FuzzySet(-10, 0, 0, 10);    // Triangular
FuzzySet *derivativePositive = new FuzzySet(0, 80, 160, 160); // Trapezoidal
derivative->addFuzzySet(derivativeNegative);
derivative->addFuzzySet(derivativeZero);
derivative->addFuzzySet(derivativePositive);
fuzzy->addFuzzyInput(derivative);

// Output variable: motor speed
FuzzyOutput *motorSpeed = new FuzzyOutput(1);
FuzzySet *slow = new FuzzySet(0, 0, 50, 100);
FuzzySet *medium = new FuzzySet(50, 100, 100, 150);
FuzzySet *fast = new FuzzySet(100, 150, 200, 200);
motorSpeed->addFuzzySet(slow);
motorSpeed->addFuzzySet(medium);
motorSpeed->addFuzzySet(fast);
fuzzy->addFuzzyOutput(motorSpeed);

// Adding rules
FuzzyRuleAntecedent *antecedent1 = new FuzzyRuleAntecedent();
antecedent1->joinWithAND(negative, derivativeNegative);
FuzzyRuleConsequent *consequent1 = new FuzzyRuleConsequent();
consequent1->addOutput(medium);
FuzzyRule *fuzzyRule1 = new FuzzyRule(1, antecedent1, consequent1);
fuzzy->addFuzzyRule(fuzzyRule1);

```

```
FuzzyRuleAntecedent *antecedent2 = new FuzzyRuleAntecedent();  
antecedent2->joinWithAND(negative, derivativeZero);  
FuzzyRuleConsequent *consequent2 = new FuzzyRuleConsequent();  
consequent2->addOutput(medium);  
FuzzyRule *fuzzyRule2 = new FuzzyRule(2, antecedent2, consequent2);  
fuzzy->addFuzzyRule(fuzzyRule2);
```

```
FuzzyRuleAntecedent *antecedent3 = new FuzzyRuleAntecedent();  
antecedent3->joinWithAND(negative, derivativePositive);  
FuzzyRuleConsequent *consequent3 = new FuzzyRuleConsequent();  
consequent3->addOutput(fast);  
FuzzyRule *fuzzyRule3 = new FuzzyRule(3, antecedent3, consequent3);  
fuzzy->addFuzzyRule(fuzzyRule3);
```

```
FuzzyRuleAntecedent *antecedent4 = new FuzzyRuleAntecedent();  
antecedent4->joinWithAND(zeroError, derivativeNegative);  
FuzzyRuleConsequent *consequent4 = new FuzzyRuleConsequent();  
consequent4->addOutput(slow);  
FuzzyRule *fuzzyRule4 = new FuzzyRule(4, antecedent4, consequent4);  
fuzzy->addFuzzyRule(fuzzyRule4);
```

```
FuzzyRuleAntecedent *antecedent5 = new FuzzyRuleAntecedent();  
antecedent5->joinWithAND(zeroError, derivativeZero);
```

```
FuzzyRuleConsequent *consequent5 = new FuzzyRuleConsequent();  
consequent5->addOutput(slow);  
FuzzyRule *fuzzyRule5 = new FuzzyRule(5, antecedent5, consequent5);  
fuzzy->addFuzzyRule(fuzzyRule5);
```

```
FuzzyRuleAntecedent *antecedent6 = new FuzzyRuleAntecedent();  
antecedent6->joinWithAND(zeroError, derivativePositive);  
FuzzyRuleConsequent *consequent6 = new FuzzyRuleConsequent();  
consequent6->addOutput(medium);  
FuzzyRule *fuzzyRule6 = new FuzzyRule(6, antecedent6, consequent6);  
fuzzy->addFuzzyRule(fuzzyRule6);
```

```
FuzzyRuleAntecedent *antecedent7 = new FuzzyRuleAntecedent();  
antecedent7->joinWithAND(positive, derivativeNegative);  
FuzzyRuleConsequent *consequent7 = new FuzzyRuleConsequent();  
consequent7->addOutput(medium);  
FuzzyRule *fuzzyRule7 = new FuzzyRule(7, antecedent7, consequent7);  
fuzzy->addFuzzyRule(fuzzyRule7);
```

```
FuzzyRuleAntecedent *antecedent8 = new FuzzyRuleAntecedent();  
antecedent8->joinWithAND(positive, derivativeZero);  
FuzzyRuleConsequent *consequent8 = new FuzzyRuleConsequent();  
consequent8->addOutput(medium);  
FuzzyRule *fuzzyRule8 = new FuzzyRule(8, antecedent8, consequent8);
```

```
fuzzy->addFuzzyRule(fuzzyRule8);

FuzzyRuleAntecedent *antecedent9 = new FuzzyRuleAntecedent();
antecedent9->joinWithAND(positive, derivativePositive);

FuzzyRuleConsequent *consequent9 = new FuzzyRuleConsequent();
consequent9->addOutput(fast);

FuzzyRule *fuzzyRule9 = new FuzzyRule(9, antecedent9, consequent9);
fuzzy->addFuzzyRule(fuzzyRule9);
}

// Setup function
void setup() {
    Serial.begin(115200);
    pixy.init();
    setupMotors();
    setupFuzzyLogic();
}

// Movement functions
void moveForward(int speed) {
    digitalWrite(IN1, HIGH);
    digitalWrite(IN3, LOW);
    analogWrite(ENA, speed);
    digitalWrite(IN2, LOW);
}
```

```
    digitalWrite(IN4, HIGH);  
    analogWrite(ENB, speed);  
}
```

```
void moveLeft(int speed) {  
    digitalWrite(IN1, LOW);  
    digitalWrite(IN3, LOW);  
    analogWrite(ENA, speed);  
    digitalWrite(IN2, HIGH);  
    digitalWrite(IN4, HIGH);  
    analogWrite(ENB, speed);  
}
```

```
void moveRight(int speed) {  
    digitalWrite(IN1, HIGH);  
    digitalWrite(IN3, HIGH);  
    analogWrite(ENA, speed);  
    digitalWrite(IN2, LOW);  
    digitalWrite(IN4, LOW);  
    analogWrite(ENB, speed);  
}
```

```
void Stop() {  
    digitalWrite(IN1, LOW);
```



```

    digitalWrite(IN3, LOW);

    analogWrite(ENA, 0);

    digitalWrite(IN2, LOW);

    digitalWrite(IN4, LOW);

    analogWrite(ENB, 0);

}

// Main loop function

void loop() {

    pixy.ccc.getBlocks();

    if (pixy.ccc.numBlocks) {

        int x = pixy.ccc.blocks[0].m_x;

        int errorValue = x - 160; // 160 is the center of the image

        int derivativeValue = errorValue - previous_error;

        // Calculate speed using fuzzy logic

        fuzzy->setInput(1, errorValue);

        fuzzy->setInput(2, derivativeValue);

        fuzzy->fuzzify();

        float motorSpeedValue = fuzzy->defuzzify(1);

        int speed = constrain(motorSpeedValue, 0, MAX_SPEED);

        // Apply fuzzy logic control to the robot's movement
    }
}

```

```
    if (errorValue < -CENTER_WIDTH) {
        moveLeft(speed);
    } else if (errorValue > CENTER_WIDTH) {
        moveRight(speed);
    } else {
        moveForward(MAX_SPEED);
    }
    previous_error = errorValue;

    // Display speeds in the serial monitor
    if (millis() - lastUpdateTime >= 1000) {
        Serial.print("Speed (Fuzzy Logic): ");
        Serial.println(speed);
        lastUpdateTime = millis();
    }

} else {
    Stop(); // Stop if no target is found
}
}
```

Appendix D – Source Code(ESP32 Cam Program)

```
/* Edge Impulse Arduino examples
 * Copyright (c) 2022 EdgeImpulse Inc.
 *
 * Permission is hereby granted, free of charge, to any person obtaining a copy
 * of this software and associated documentation files (the "Software"), to deal
 * in the Software without restriction, including without limitation the rights
 * to use, copy, modify, merge, publish, distribute, sublicense, and/or sell
 * copies of the Software, and to permit persons to whom the Software is
 * furnished to do so, subject to the following conditions:
 *
 * The above copyright notice and this permission notice shall be included in
 * all copies or substantial portions of the Software.
 *
 * THE SOFTWARE IS PROVIDED "AS IS", WITHOUT WARRANTY OF ANY
KIND, EXPRESS OR
 * IMPLIED, INCLUDING BUT NOT LIMITED TO THE WARRANTIES OF
MERCHANTABILITY,
 * FITNESS FOR A PARTICULAR PURPOSE AND NONINFRINGEMENT. IN NO
EVENT SHALL THE
 * AUTHORS OR COPYRIGHT HOLDERS BE LIABLE FOR ANY CLAIM,
DAMAGES OR OTHER
 * LIABILITY, WHETHER IN AN ACTION OF CONTRACT, TORT OR
OTHERWISE, ARISING FROM,
 * OUT OF OR IN CONNECTION WITH THE SOFTWARE OR THE USE OR
OTHER DEALINGS IN THE
 * SOFTWARE.
 */

// These sketches are tested with 2.0.4 ESP32 Arduino Core
// https://github.com/espressif/arduino-esp32/releases/tag/2.0.4

/* Includes ----- */
```

```
#include <SEK.SOUN-project-1_inferencing.h>
#include "edge-impulse-sdk/dsp/image/image.hpp"

#include "esp_camera.h"

// Select camera model - find more camera models in camera_pins.h file here
// https://github.com/espressif/arduino-esp32/blob/master/libraries/ESP32/examples/Camera/CameraWebServer/camera_pins.h

#ifndef CAMERA_MODEL_ESP_EYE // Has PSRAM
#define CAMERA_MODEL_AI_THINKER // Has PSRAM

#if defined(CAMERA_MODEL_ESP_EYE)
#define PWDN_GPIO_NUM -1
#define RESET_GPIO_NUM -1
#define XCLK_GPIO_NUM 4
#define SIOD_GPIO_NUM 18
#define SIOC_GPIO_NUM 23

#define Y9_GPIO_NUM 36
#define Y8_GPIO_NUM 37
#define Y7_GPIO_NUM 38
#define Y6_GPIO_NUM 39
#define Y5_GPIO_NUM 35
#define Y4_GPIO_NUM 14
#define Y3_GPIO_NUM 13
#define Y2_GPIO_NUM 34
#define VSYNC_GPIO_NUM 5
#define HREF_GPIO_NUM 27
#define PCLK_GPIO_NUM 25

#elif defined(CAMERA_MODEL_AI_THINKER)
```

```
#define PWDN_GPIO_NUM 32
#define RESET_GPIO_NUM -1
#define XCLK_GPIO_NUM 0
#define SIOD_GPIO_NUM 26
#define SIOC_GPIO_NUM 27
```

```
#define Y9_GPIO_NUM 35
#define Y8_GPIO_NUM 34
#define Y7_GPIO_NUM 39
#define Y6_GPIO_NUM 36
#define Y5_GPIO_NUM 21
#define Y4_GPIO_NUM 19
#define Y3_GPIO_NUM 18
#define Y2_GPIO_NUM 5
#define VSYNC_GPIO_NUM 25
#define HREF_GPIO_NUM 23
#define PCLK_GPIO_NUM 22
```

```
#else
#error "Camera model not selected"
#endif
```

```
/* Constant defines ----- */
```

```
#define EI_CAMERA_RAW_FRAME_BUFFER_COLS 320
#define EI_CAMERA_RAW_FRAME_BUFFER_ROWS 240
#define EI_CAMERA_FRAME_BYTE_SIZE 3
```

```
/* Private variables ----- */
```

```
static bool debug_nn = false; // Set this to true to see e.g. features generated from the raw
signal
static bool is_initialised = false;
uint8_t *snapshot_buf; //points to the output of the capture
```

```

static camera_config_t camera_config = {
    .pin_pwdn = PWDN_GPIO_NUM,
    .pin_reset = RESET_GPIO_NUM,
    .pin_xclk = XCLK_GPIO_NUM,
    .pin_sscb_sda = SIOD_GPIO_NUM,
    .pin_sscb_scl = SIOC_GPIO_NUM,

    .pin_d7 = Y9_GPIO_NUM,
    .pin_d6 = Y8_GPIO_NUM,
    .pin_d5 = Y7_GPIO_NUM,
    .pin_d4 = Y6_GPIO_NUM,
    .pin_d3 = Y5_GPIO_NUM,
    .pin_d2 = Y4_GPIO_NUM,
    .pin_d1 = Y3_GPIO_NUM,
    .pin_d0 = Y2_GPIO_NUM,
    .pin_vsync = VSYNC_GPIO_NUM,
    .pin_href = HREF_GPIO_NUM,
    .pin_pclk = PCLK_GPIO_NUM,

    //XCLK 20MHz or 10MHz for OV2640 double FPS (Experimental)
    .xclk_freq_hz = 20000000,
    .ledc_timer = LEDC_TIMER_0,
    .ledc_channel = LEDC_CHANNEL_0,

    .pixel_format = PIXFORMAT_JPEG, //YUV422,GRAYSCALE,RGB565,JPEG
    .frame_size = FRAMESIZE_QVGA, //QQVGA-UXGA Do not use sizes above
    QVGA when not JPEG

    .jpeg_quality = 12, //0-63 lower number means higher quality
    .fb_count = 1, //if more than one, i2s runs in continuous mode. Use only with JPEG
    .fb_location = CAMERA_FB_IN_PSRAM,

```

```

        .grab_mode = CAMERA_GRAB_WHEN_EMPTY,
    };

/* Function definitions ----- */
bool ei_camera_init(void);
void ei_camera_deinit(void);
bool ei_camera_capture(uint32_t img_width, uint32_t img_height, uint8_t *out_buf) ;

/**
 * @brief   Arduino setup function
 */
void setup()
{
    // put your setup code here, to run once:
    Serial.begin(115200);
    //comment out the below line to start inference immediately after upload
    while (!Serial);
    Serial.println("Edge Impulse Inferencing Demo");
    if (ei_camera_init() == false) {
        ei_printf("Failed to initialize Camera!\r\n");
    }
    else {
        ei_printf("Camera initialized\r\n");
    }

    ei_printf("\nStarting continious inference in 2 seconds...\n");
    ei_sleep(2000);
}

/**
 * @brief   Get data and run inferencing
 */

```

```

* @param[in] debug Get debug info if true
*/
void loop()
{

    // instead of wait_ms, we'll wait on the signal, this allows threads to cancel us...
    if (ei_sleep(5) != EI_IMPULSE_OK) {
        return;
    }

    snapshot_buf = (uint8_t*)malloc(EI_CAMERA_RAW_FRAME_BUFFER_COLS *
EI_CAMERA_RAW_FRAME_BUFFER_ROWS * EI_CAMERA_FRAME_BYTE_SIZE);

    // check if allocation was successful
    if(snapshot_buf == nullptr) {
        ei_printf("ERR: Failed to allocate snapshot buffer!\n");
        return;
    }

    ei::signal_t signal;
    signal.total_length = EI_CLASSIFIER_INPUT_WIDTH *
EI_CLASSIFIER_INPUT_HEIGHT;
    signal.get_data = &ei_camera_get_data;

    if (ei_camera_capture((size_t)EI_CLASSIFIER_INPUT_WIDTH,
(size_t)EI_CLASSIFIER_INPUT_HEIGHT, snapshot_buf) == false) {
        ei_printf("Failed to capture image\r\n");
        free(snapshot_buf);
        return;
    }

    // Run the classifier
    ei_impulse_result_t result = { 0 };

```



```
    EI_IMPULSE_ERROR err = run_classifier(&signal, &result, debug_nn);
    if (err != EI_IMPULSE_OK) {
        ei_printf("ERR: Failed to run classifier (%d)\n", err);
        return;
    }

    // print the predictions
    ei_printf("Predictions (DSP: %d ms., Classification: %d ms., Anomaly: %d ms.): \n",
              result.timing.dsp, result.timing.classification, result.timing.anomaly);

#ifdef EI_CLASSIFIER_OBJECT_DETECTION == 1
    ei_printf("Object detection bounding boxes:\r\n");
    for (uint32_t i = 0; i < result.bounding_boxes_count; i++) {
        ei_impulse_result_bounding_box_t bb = result.bounding_boxes[i];
        if (bb.value == 0) {
            continue;
        }
        ei_printf(" %s (%f) [ x: %u, y: %u, width: %u, height: %u ]\r\n",
                  bb.label,
                  bb.value,
                  bb.x,
                  bb.y,
                  bb.width,
                  bb.height);
    }

    // Print the prediction results (classification)
#else
    ei_printf("Predictions:\r\n");
    for (uint16_t i = 0; i < EI_CLASSIFIER_LABEL_COUNT; i++) {
        ei_printf(" %s: ", ei_classifier_inferencing_categories[i]);
    }
#endif
```

```
        ei_printf("%.5f\r\n", result.classification[i].value);
    }
#endif

    // Print anomaly result (if it exists)
#ifdef EI_CLASSIFIER_HAS_ANOMALY
    ei_printf("Anomaly prediction: %.3f\r\n", result.anomaly);
#endif

#ifdef EI_CLASSIFIER_HAS_VISUAL_ANOMALY
    ei_printf("Visual anomalies:\r\n");
    for (uint32_t i = 0; i < result.visual_ad_count; i++) {
        ei_impulse_result_bounding_box_t bb = result.visual_ad_grid_cells[i];
        if (bb.value == 0) {
            continue;
        }
        ei_printf(" %s (%f) [ x: %u, y: %u, width: %u, height: %u ]\r\n",
            bb.label,
            bb.value,
            bb.x,
            bb.y,
            bb.width,
            bb.height);
    }
#endif

    free(snapshot_buf);

}

/**
```

```

* @brief Setup image sensor & start streaming
*
* @retval false if initialisation failed
*/
bool ei_camera_init(void) {

    if (is_initialised) return true;

#ifdef CAMERA_MODEL_ESP_EYE
    pinMode(13, INPUT_PULLUP);
    pinMode(14, INPUT_PULLUP);
#endif

    //initialize the camera
    esp_err_t err = esp_camera_init(&camera_config);
    if (err != ESP_OK) {
        Serial.printf("Camera init failed with error 0x%x\n", err);
        return false;
    }

    sensor_t * s = esp_camera_sensor_get();
    // initial sensors are flipped vertically and colors are a bit saturated
    if (s->id.PID == OV3660_PID) {
        s->set_vflip(s, 1); // flip it back
        s->set_brightness(s, 1); // up the brightness just a bit
        s->set_saturation(s, 0); // lower the saturation
    }

#ifdef CAMERA_MODEL_M5STACK_WIDE
    s->set_vflip(s, 1);
    s->set_hmirror(s, 1);
#endif
#ifdef CAMERA_MODEL_ESP_EYE

```

```

s->set_vflip(s, 1);
s->set_hmirror(s, 1);
s->set_awb_gain(s, 1);
#endif

    is_initialised = true;
    return true;
}

/**
 * @brief    Stop streaming of sensor data
 */
void ei_camera_deinit(void) {

    //deinitialize the camera
    esp_err_t err = esp_camera_deinit();

    if (err != ESP_OK)
    {
        ei_printf("Camera deinit failed\n");
        return;
    }

    is_initialised = false;
    return;
}

/**
 * @brief    Capture, rescale and crop image
 *
 * @param[in] img_width    width of output image

```

```

* @param[in] img_height  height of output image
* @param[in] out_buf     pointer to store output image, NULL may be used
*
*                       if ei_camera_frame_buffer is to be used for capture and
resize/cropping.
*
* @retval  false if not initialised, image captured, rescaled or cropped failed
*
*/
bool ei_camera_capture(uint32_t img_width, uint32_t img_height, uint8_t *out_buf) {
    bool do_resize = false;

    if (!is_initialised) {
        ei_printf("ERR: Camera is not initialized\r\n");
        return false;
    }

    camera_fb_t *fb = esp_camera_fb_get();

    if (!fb) {
        ei_printf("Camera capture failed\n");
        return false;
    }

    bool converted = fmt2rgb888(fb->buf, fb->len, PIXFORMAT_JPEG, snapshot_buf);

    esp_camera_fb_return(fb);

    if(!converted){
        ei_printf("Conversion failed\n");
        return false;
    }
}

```

```

if ((img_width != EI_CAMERA_RAW_FRAME_BUFFER_COLS)
    || (img_height != EI_CAMERA_RAW_FRAME_BUFFER_ROWS)) {
    do_resize = true;
}

if (do_resize) {
    ei::image::processing::crop_and_interpolate_rgb888(
        out_buf,
        EI_CAMERA_RAW_FRAME_BUFFER_COLS,
        EI_CAMERA_RAW_FRAME_BUFFER_ROWS,
        out_buf,
        img_width,
        img_height);
}

return true;
}

static int ei_camera_get_data(size_t offset, size_t length, float *out_ptr)
{
    // we already have a RGB888 buffer, so recalculate offset into pixel index
    size_t pixel_ix = offset * 3;
    size_t pixels_left = length;
    size_t out_ptr_ix = 0;

    while (pixels_left != 0) {
        // Swap BGR to RGB here
        // due to https://github.com/espressif/esp32-camera/issues/379
        out_ptr[out_ptr_ix] = (snapshot_buf[pixel_ix + 2] << 16) + (snapshot_buf[pixel_ix
+ 1] << 8) + snapshot_buf[pixel_ix];
    }
}

```

```
    // go to the next pixel
    out_ptr_ix++;
    pixel_ix+=3;
    pixels_left--;
}
// and done!
return 0;
}
```

```
    #if !defined(EI_CLASSIFIER_SENSOR) || EI_CLASSIFIER_SENSOR !=
EI_CLASSIFIER_SENSOR_CAMERA
```

```
        #error "Invalid model for current sensor"
```

```
    #endif
```

Appendix E – Source Code(ESP32 –Arduino-PID)

```
```.cpp

// Define motor pins on the L298N module

const int ENA = 5;

const int IN1 = 7;

const int IN2 = 6;

const int ENB = 10;

const int IN3 = 9;

const int IN4 = 8;

// Maximum movement speed and central zone width

const int MAX_SPEED = 200;

const int CENTER_WIDTH = 40;

int speed;

// PID variables

float Kp = 0.2; // Proportional constant

float Ki = 0.01; // Integral constant

float Kd = 0.1; // Derivative constant

int previous_error = 0;

float integral = 0;

// Variable for target position (x)

int x = 0;
```



```
// Motor setup

void setupMotors() {

 pinMode(ENA, OUTPUT);

 pinMode(IN1, OUTPUT);

 pinMode(IN2, OUTPUT);

 pinMode(ENB, OUTPUT);

 pinMode(IN3, OUTPUT);

 pinMode(IN4, OUTPUT);

}
```

```
// Setup function
```

```
void setup() {

 Serial.begin(115200);

 setupMotors();

}
```

```
// Move forward function
```

```
void moveForward(int speed) {

 Serial.println("Forward.");

 digitalWrite(IN1, HIGH);

 digitalWrite(IN3, LOW);

 analogWrite(ENA, speed);

 digitalWrite(IN2, LOW);

}
```

```
 digitalWrite(IN4, HIGH);
 analogWrite(ENB, speed);
}
```

```
// Move left function
```

```
void moveLeft(int speed) {
 Serial.println("Left.");
 digitalWrite(IN1, LOW);
 digitalWrite(IN3, LOW);
 analogWrite(ENA, speed);
 digitalWrite(IN2, HIGH);
 digitalWrite(IN4, HIGH);
 analogWrite(ENB, speed);
}
```

```
// Move right function
```

```
void moveRight(int speed) {
 Serial.println("Right.");
 digitalWrite(IN1, HIGH);
 digitalWrite(IN3, HIGH);
 analogWrite(ENA, speed);
 digitalWrite(IN2, LOW);
 digitalWrite(IN4, LOW);
 analogWrite(ENB, speed);
}
```

```
 }

 // Stop function
 void Stop() {
 Serial.println("Stop.");
 digitalWrite(IN1, LOW);
 digitalWrite(IN3, LOW);
 analogWrite(ENA, 0);
 digitalWrite(IN2, LOW);
 digitalWrite(IN4, LOW);
 analogWrite(ENB, 0);
 }

 // Main loop function
 void loop() {
 if (Serial.available()) {
 String receivedMessage = Serial.readStringUntil('\n'); // Read data until end of line
 receivedMessage.trim(); // Remove excess spaces (including '\n')

 // Display received data on Serial Monitor
 Serial.print("Received data from ESP32: ");
 Serial.println(receivedMessage);

 // Find the position of the word "x: "
```

```

int xPosIndex = receivedMessage.indexOf("x: ");

if (xPosIndex != -1) { // If "x: " is found

 // Extract the part of the text starting from "x: " to the end of the following
number
 String xPosStr = receivedMessage.substring(xPosIndex + 3); // "+3" to skip "x: "

 // Convert the text to an integer

 x = xPosStr.toInt();

 // Display xPos value on Arduino Serial Monitor

 Serial.print("Position of scorpion (X): ");

 Serial.println(x);

 // Calculate PID and control movement based on target position

 int error = x - 30; // 30 is the center in the image

 integral += error;

 float derivative = error - previous_error;

 speed = Kp * error + Ki * integral + Kd * derivative; // Calculate PID

 speed = constrain(speed, 0, MAX_SPEED); // Ensure speed is within allowed
limits

 if (error < -CENTER_WIDTH) {

 moveLeft(speed);

 } else if (error > CENTER_WIDTH) {

 moveRight(speed);

```

```
 } else {
 moveForward(MAX_SPEED);
 }

 previous_error = error; // Update previous error for next time
} else {
 // If "x: " is not found, the robot stops
 Stop();
}
}
}
```

## Appendix \_Permission to print

الجمهورية الجزائرية الديمقراطية الشعبية  
République Algérienne Démocratique et Populaire  
وزارة التعليم العالي والبحث العلمي  
Ministère de l'Enseignement Supérieur Et de La Recherche Scientifique  
جامعة غرداية



Université de Ghardaïa

Faculté des sciences et Technologies  
Département d'automatique et  
d'électromécanique

كلية العلوم والتكنولوجيا  
قسم الآلية والكهروميكانيك

غرداية في: 2024/09/15

### إذن بالطباعة (مذكرة ماستر)

بعد الاطلاع على التصحيحات المطلوبة على محتوى المذكرة المنجزة من طرف الطلبة التالية أسماؤهم:

1. الطالب (ة): صغير سندس
2. الطالب (ة): /

تخصص: آلية وأنظمة  
نمنح نحن أعضاء لجنة المناقشة:

الإمضاء	الصفة	المؤسسة الأصلية	الرتبة	الإسم واللقب
	الممتحن 1	جامعة غرداية	MCB	مصباح سعيد
/	الممتحن 2	/	/	/
	المؤطر	جامعة غرداية	MCA	حسن ناصر
	رئيس اللجنة	جامعة غرداية	MCB	شوية فيصل

الإذن بطباعة النسخة النهائية لمذكرة الماستر الموسومة بعنوان:

**Robotic Automated Scorpion and Snake Venom Extraction System**

إمضاء رئيس القسم

العلمي عبد اللطيف  
رئيس قسم الآلية  
والكهروميكانيك

