Minister of Higher Education, Scientific Research,
and Information and Communication Technology.

Higher Institute of Technological Studies of Béja

Department of Computer Technology

***Graduation Project Report submitted to obtain the degree of***

National Diploma of License in Development of Embedded and Mobile Systems

# Design and implementation of a system for detecting non-compliances in a product in the production line

Defended on –/–/2023 in front of the committee composed of:

- Last name and first name of member 1, Grade, Affiliation (President)
- Last name and first name of member 2, Grade, Affiliation (Reviewer)
- Last name and first name of member 3, Grade, Affiliation (Company supervisor)
- Last name and first name of member 4, Grade, Affiliation (University supervisor)

**End of studies internship completed at**

**Host company** 


**Client company** 


Academic year 2022-2023

# Thanks

As I sit down to pen my farewell message at the end of my final year's internship, my mind is filled with a swirl of emotions. I can't help but feel a deep sense of gratitude towards the people who have helped me grow and develop over the past few months. Foremost among them is my supervisor, Mr Radhwen Aloui. His guidance, support, and mentorship have been invaluable to me, and I am truly grateful for the opportunity to have worked under his tutelage. His constant feedback and encouragement have helped me push my limits and reach new heights, particularly in the field of AI, where he encouraged me to excel and explore my potential.

I would also like to extend my thanks to my academic institution, where I have spent the past three years studying and learning. The institution's safe and educative environment has provided me with a solid foundation on which I can build my future. I will always cherish the memories of my time here, the friends I made, and the knowledge I gained.

Lastly, I would like to express my heartfelt gratitude to the jury members who took the time to review my rapport. I understand that your time is precious, and I appreciate your efforts in evaluating my work. I hope that my report reflects the hard work and dedication that I have put into my project over the past four months. Once again, thank you to everyone who has contributed to my growth and success during my internship, and I will always cherish the lessons and experiences that I have gained from this remarkable journey.

# Contents

# List of Figures

# List of Tables

# Introduction

The lack of precision in manufacturing has been a persistent challenge for businesses of all sizes, from the largest manufacturing plants to the most humble workshops. Manufacturing businesses are often faced with the challenge of maintaining consistent precision in their production processes. The slightest deviation from the required standards can result in defective products, which can cause harm to the business's reputation and decrease sales. The causes of lack of precision can be numerous, ranging from technical glitches in production equipment to human error in the assembly line. Furthermore, production inefficiencies can also lead to a lack of precision, as poorly designed or executed manufacturing processes can result in inconsistent products.[1]

In addition to the negative consequences of producing defective products, the costs associated with reworking, scrapping, or returning products can add up quickly. These expenses can negatively impact the company's bottom line and can even jeopardize the company's financial stability in the long run. As a result, it is critical for manufacturing businesses to implement rigorous quality control measures to ensure that products meet or exceed the required standards consistently. [2]

One solution that has shown great promise is the use of artificial intelligence and machine learning algorithms, such as neural networks, to optimize production processes and improve product quality.[1]

Neural networks and other artificial intelligence technologies have the potential to revolutionize the manufacturing industry by enabling businesses to identify and address issues more quickly and efficiently. With the ability to analyze vast amounts of data in real-time, these technologies can help businesses make informed decisions about how to improve their processes and reduce the risk of defective products. Moreover, by reducing the costs associated with reworking, scrapping, or returning products, these technologies can also help businesses improve their bottom line and increase profitability.[1][3]

At TESCA, the company where I completed my internship, we were able to leverage the power of YOLOV5 to mitigate the problem of defective products and returns. By analyzing vast amounts of data from our manufacturing processes, we were able to identify patterns and correlations that were not immediately visible to the human eye. This enabled us to make products that were consistently of higher quality and closer to the required standards.

In conclusion, while the lack of precision in manufacturing remains a persistent challenge for many businesses, advances in technology offer promising solutions. By leveraging the power of artificial intelligence and machine learning algorithms, such as neural networks, businesses can gain deeper insights into their production processes, optimize their operations, and improve the quality of their products. As a result, they can reduce the risk of defective products and associated costs, while also improving their reputation and bottom line.

# Chapter 1

# Project Framework and Methodology

## I Introduction

The first chapter of this report aims to provide an overview of the host company, present a case study highlighting the problem to be addressed, and introduce the modeling language used for the project. This chapter sets the stage for the subsequent sections by establishing the context, significance, and scope of the actions undertaken in the next chapters.

## II Presentation of the enterprise

Tesca Group is a family-run company with a background in textiles that has successfully ventured into the automotive industry, specifically automotive seating. Emulating the approach of a French automotive manufacturing house, Tesca Group meticulously crafts each piece, from its atelier to its global production sites[26]. The company strives to embody the same creative audacity and prides itself on being a partner to major automotive manufacturers like :


Figure 1.1: Tesca's partners

With a clientele consisting of demanding visionaries who shape the future of mobility, Tesca Group positions itself as a reliable and flexible partner, accompanying its clients in their pursuit of innovation and competitiveness. The company believes that adopting a long-term and eco-friendly vision is crucial for sustainable growth, constantly pushing for excellence rather than settling for mediocrity.[26]

Sustainability is a concrete and integral part of Tesca Group's approach. The company is committed to sustainable development and actively incorporates environmental and social values into its products, services, and activities within the automotive industry. Since 2004, Tesca Group has been dedicated to controlling its ecological footprint, and the executive management maintains and expands upon these efforts. The company adheres to the values and principles of the Global Compact through its Ethics Charter, with the objective of obtaining ISO 14001 certification for all its sites.[26]

Tesca Group's environmental policy revolves around five major directions: behaving in an environmentally responsible manner, optimizing and controlling industrial waste, optimizing energy consumption and natural resources, participating in the reduction of the ecological footprint throughout the design and production processes, and ensuring compliance with environmental regulations and requirements.[26]

Innovation is a driving force within Tesca Group, with a constant emphasis on generating new ideas and executing them effectively. The company's research programs focus on key themes in the automotive industry, such as weight reduction, optimized part design, and ecological footprint expertise, while prioritizing user comfort, functionality, and safety. Tesca Group holds numerous patents in the conception, design, and manufacturing of automotive parts and seat components, including headrests, armrests, upholstery, padding, and "smart textiles."[26]

Industrial excellence is a cornerstone of Tesca Group's strategy, whether through its local production sites or research centers. The company strives to provide reliable processes that meet the evolving market expectations worldwide. Tesca Group empowers its teams through a culture of continuous improvement, with competitiveness, efficiency, and exceptional service forming the foundation of its industrial success. SPRINT (Tesca's INdustrial Production System) is employed to engage collaborators at all levels, aiming to standardize procedures and ensure perpetual progress in industrial performance. Shared values among Tesca Group employees include excellence, self-discipline, commitment, autonomy, and pragmatism.[26]

Tesca Group's Quality policy aligns with five major directions: meeting commitments to conformity, cost, and deadlines; aligning industrial and purchasing performance with the highest quality criteria; standardizing existing processes to offer innovative products; prioritizing process control and prevention; and continuously enhancing the skills of its teams.[26]

In terms of its history, Tesca Group has a notable timeline of achievements. Starting in 1960 with the provision of Citroën 2CV roofs, the company went on to pioneer and apply the Foam In Place technology in 1978. In 1983, Tesca Group established its Design Studio in Paris, followed by the creation of the CERA research and development center in Reims in 1993. By the year 2000, the company had expanded its production capabilities worldwide. In 2016, Tesca Group underwent a significant transformation, becoming Trèves TSC.[26]

Overall, Tesca Group has established itself as a reputable partner to major automotive manufacturers, driven by a commitment to sustainability, innovation, industrial excellence, and quality. Through its diverse range of products and services, the company aims to shape the future[26]

# III Case study

## 1 description of the existing system


Figure 1.2: IFM O2D220

Tesca is a factory that utilizes an object recognition sensor called IFM O2D220 to detect potential defects in logos present on its products. The IFM O2D220 employs two types of sensors: the Contour Sensor and the Pixel Counter.

The Contour Sensor is designed to quickly analyze and compare the defined shape of an object with similar objects. It is particularly useful when the shape of the inspected objects is repetitive. The sensor uses incident light or backlight to detect the contours of an object and then compares them with contours from reference images. Based on the level of conformity, the sensor generates an output indicating the presence of a model and, if applicable, identifies the specific model found.[27]

The Pixel Counter sensor analyzes the area of an object by counting the pixels. This sensor is best suited for inspecting objects that vary in terms of shape, size, or shade. It also utilizes incident light or backlight to detect object contours. By comparing the pixel count with predetermined thresholds, the sensor determines the characteristics and features of the object.[27]

To elaborate further, let's examine a couple of examples of the ifm object recognition sensor in action:

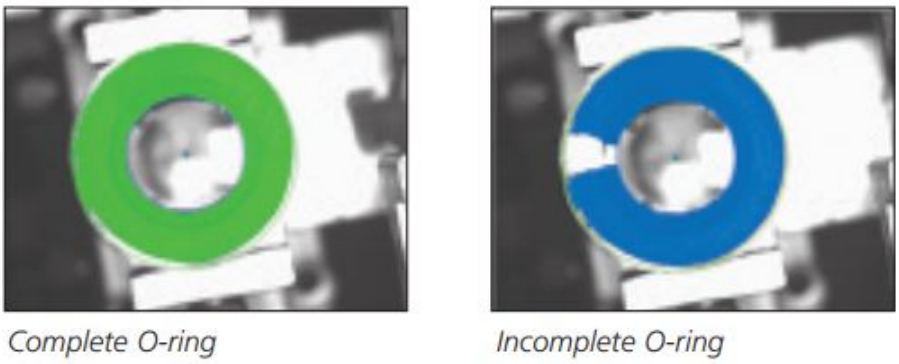



Figure 1.3: Identify missing piece in O-ring assembly with IFM O2D220

Identify missing piece in O-ring assembly : as you can see in figure 1.3, identifying that a piece is missingin an O-ring is imperative. The Pixel Counter is programmed to verify that the O-ring is complete and that no piece is missing regardless of the size and location of the missing piece. [27]

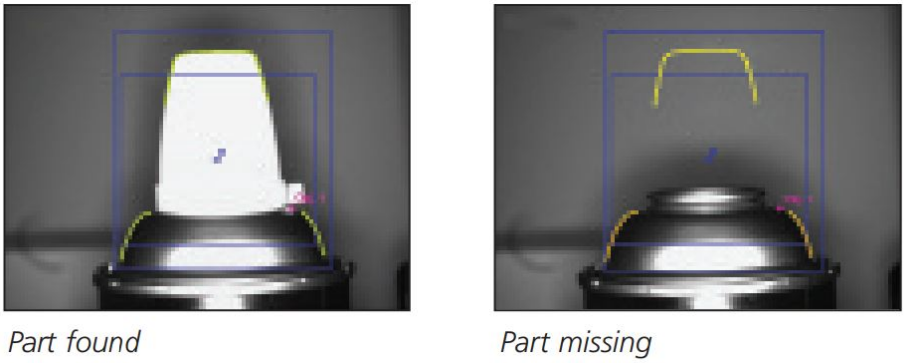



Figure 1.4: Identify cap on top of spray can with IFM O2D220

Identify cap on top of spray can : you can see in figure 1.4, The cap can be missing or improperly installed. By learning the outline of the top of the cap and the canister, the Contour Sensor allows detection of missing or incorrectly.[27] installed caps.

## 2 criticism of the existing system


Figure 1.5: Product's place holder

In the factory's workflow, workers are responsible for placing the individual pieces of the product into designated placeholders along the assembly line seen in figure 1.5 . Once the piece is securely positioned, the system relies on the ifm object recognition sensor to assess its quality. The ifm sensor analyzes the piece, evaluating its characteristics and comparing them to predetermined quality criteria. If the piece meets the required standards and is determined to be good, the system triggers an automated mechanism associated with the placeholder. As a result, the placeholder smoothly lowers its position, indicating that the piece has been approved and can proceed to the next stage of production. the problem here is that When selecting the operating distance it has to be taken into account that contour detection becomes less reliable with decreasing size of the objects. The objects to be detected should cover at least 5% of the field of view The requirement for objects to cover a certain percentage of the field of view indicates that the existing system may struggle to detect smaller objects reliably. This can result in potential issues such as false positives or missed detections, leading to a decrease in overall efficiency and quality control.

Another issue that may arise with product placement is that this soltion is based on preconfigured rules. In the given example, if we focus on Figure 1.4, it is apparent that if the normal can were to be placesd slitly above , it would mistakenly be detected as a defective piece because the the putline of the cap will be above the preconfigured outline.

The reliance on fixed rules poses a challenge when dealing with variations in product placement that fall within acceptable tolerances. In cases where slight deviations occur, such as minor misalignments or positioning errors, the system may incorrectly classify the product as defective, leading to unnecessary interruptions in the production process.

As a result, this limitation can lead to false detection of defects and potentially disrupt the production line. It highlights the importance of considering alternative or supplementary methods for accurately assessing the presence of the cap.

## 3 Solution

To address the limitations and challenges associated with the existing contour detection system, I propose an alternative solution that leverages the power of the YOLOv5 object detection. By integrating a system based on YOLOv5, we can overcome the drawbacks of the current approach and enhance our ability to detect defective logos on the production line. The improved accuracy of YOLOv5 ensures more precise identification and localization of logos, reducing the chances of false

positives or missed detections. This is especially valuable when dealing with logos of varying sizes, shapes, and orientations.

### 4 Modeling languages : SysMl

**SysMl**, or Systems Modeling Language, is a graphical language used by MBSE (Model-Based Systems Engineering) practitioners to create system models and communicate ideas about system designs to stakeholders. It is a language that has a grammar and vocabulary consisting of graphical notations that have specific meanings. The purpose of SysML is to visualize and communicate a system's design among stakeholders.[4]

The grammar and notations of SysML are defined in a standards specification published by the Object Management Group, which is a consortium of computer industry companies, government agencies, and academic institutions. SysML is an extension of a subset of the Unified Modeling Language (UML), and its complete definition requires referring to parts of the UML specification document as well.[4] We can name a few types of SysMl diagrams:

- Behavior diagrams[4]
  - Activity diagram
  - State machine diagram
  - Sequence diagram
  - Use case diagram
  - Communication diagram
- Structure diagrams[4]
  - Block definition diagram
  - Internal block diagram
  - Parametric diagram
  - Package diagram
  - Composite structure diagram
- Requirement diagram[4]

## IV Conclusion

Overall, chapter 1 has provided the necessary foundation for the subsequent chapters, enabling a comprehensive exploration of the problem and the development of an effective solution. By understanding the host company, the specific problem through the case study, and the modeling language employed, you the reader will be equipped with the context and knowledge required to delve deeper into the next chapters presented in this report.

# Chapter 2

# Logo Detection with Image Processing

## I Introduction

In this chapter, we explore the workflow involved in detecting a logo on different images captured using various devices. We begin by discussing the necessary steps required before applying contour detection to the images, including brightness and contrast adjustment, reflection removal, and grayscaling. We then examine the results of applying these techniques on three different images captured using a high-end phone camera, an OV2640 camera module, and an ESP32-CAM, respectively. Through our experiments, we observe the impact of image capture devices on image quality and demonstrate the effectiveness of image processing techniques in enhancing image quality for logo detection.

## II Specification of requirements

In this section, we introduce the functional and non-functional requirements.

### 1 Description of functional requirements

- The ESP32-CAM board should be able to capture a live video and make it available through its built-in server.
- The PC should be able to connect to the server on the ESP32-CAM board and retrieve the images or videos.
- The PC should be able to process the images to determine if a specific logo is present or not.
- The PC should be able to provide some form of feedback or output indicating whether the logo is present and if it is flipped or not.

### 2 Description of non-functional requirements

The requirements do not stop at the functional level but tend towards requirements that contribute to better quality of the application. The most important ones are:

- **Reliability:** The system should be able to consistently capture and process images accurately.
- **Performance:** The system should be able to process images quickly and without noticeable lag or delay.
- **Maintainability:** The system should be easy to maintain and update, with clear and well-documented code and configuration.

# III Modeling languages diagrams

## 1 SysMl

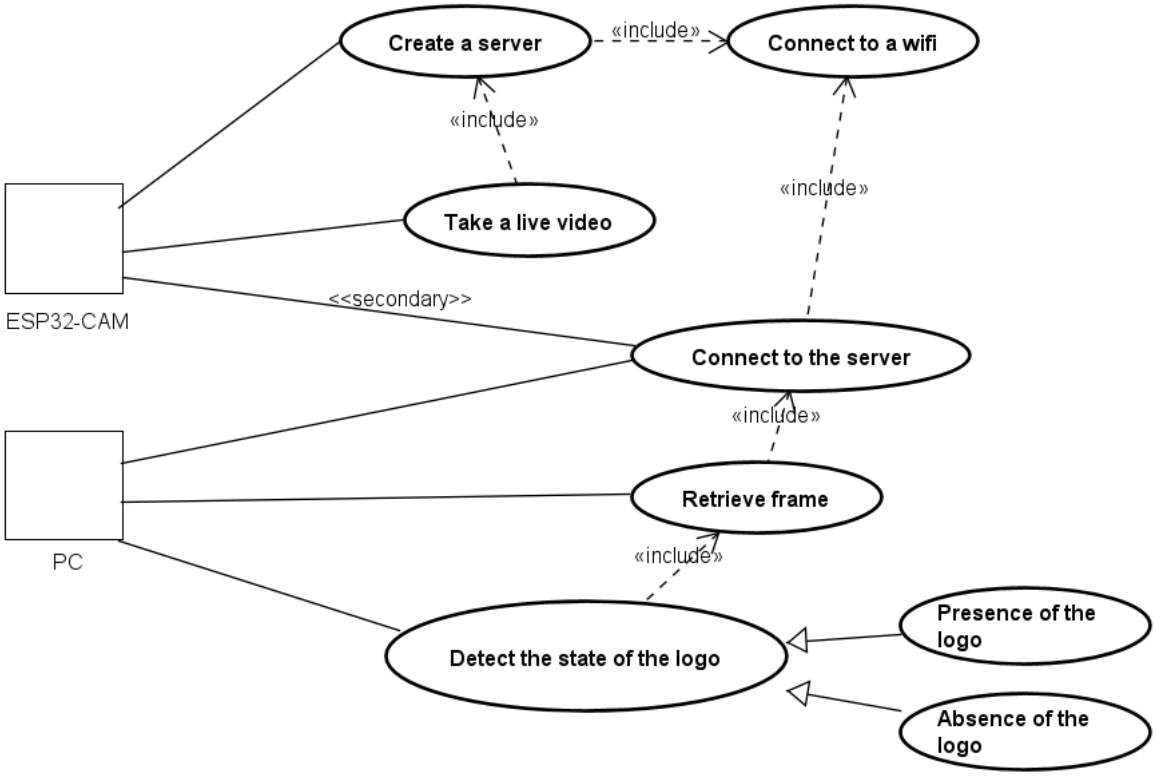



Figure 2.1: Use case diagram version 0

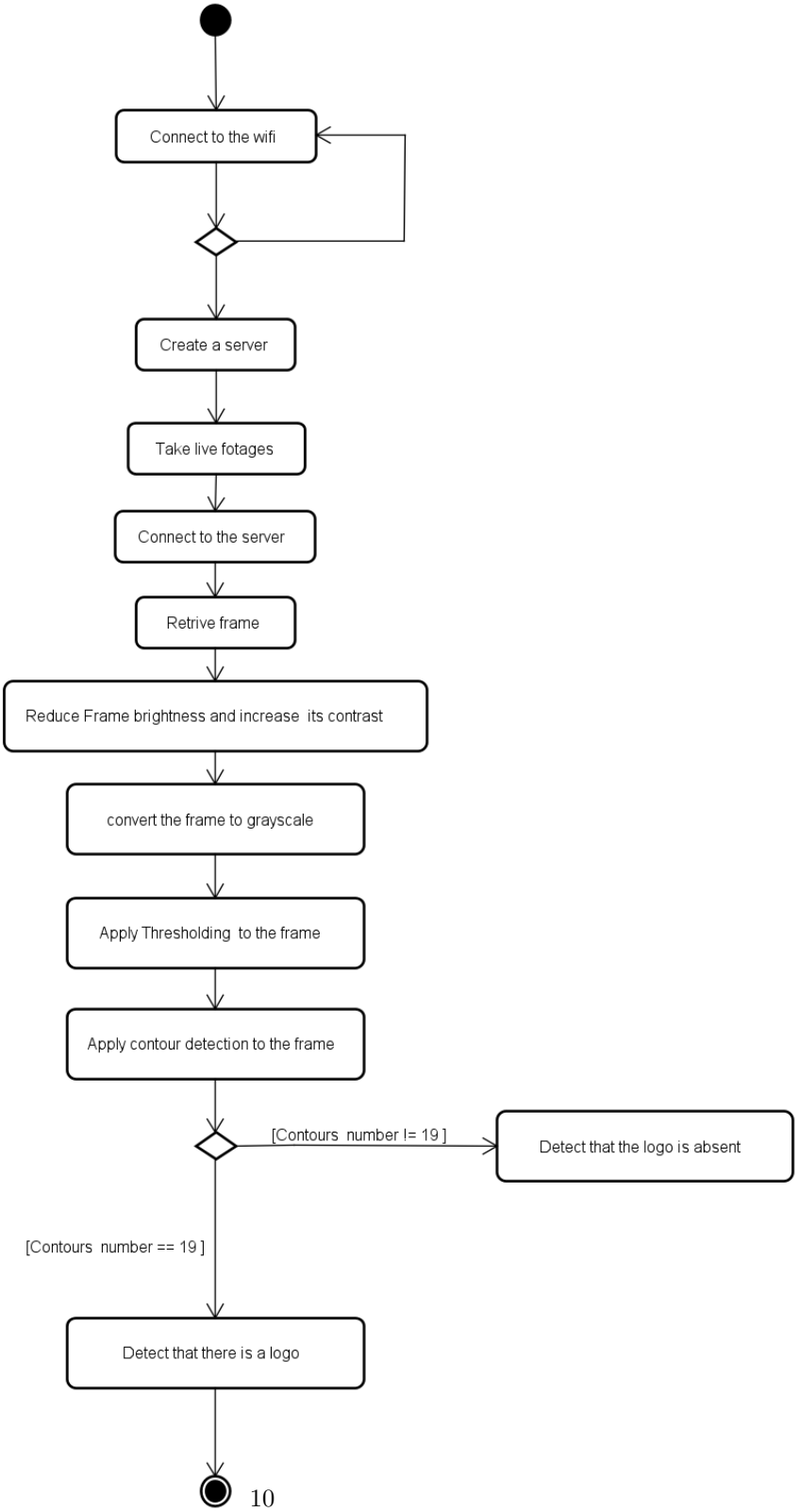





Figure 2.2: Activity diagram version 0

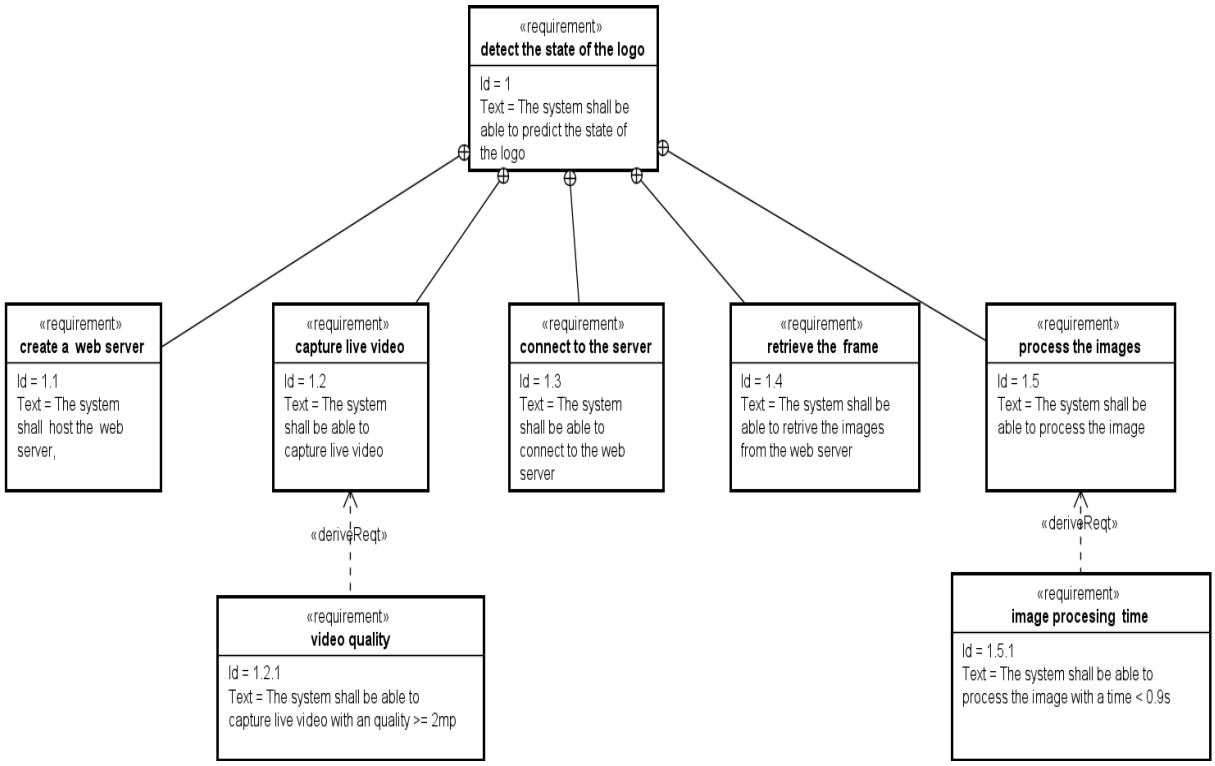



Figure 2.3: Requirement diagram version 0

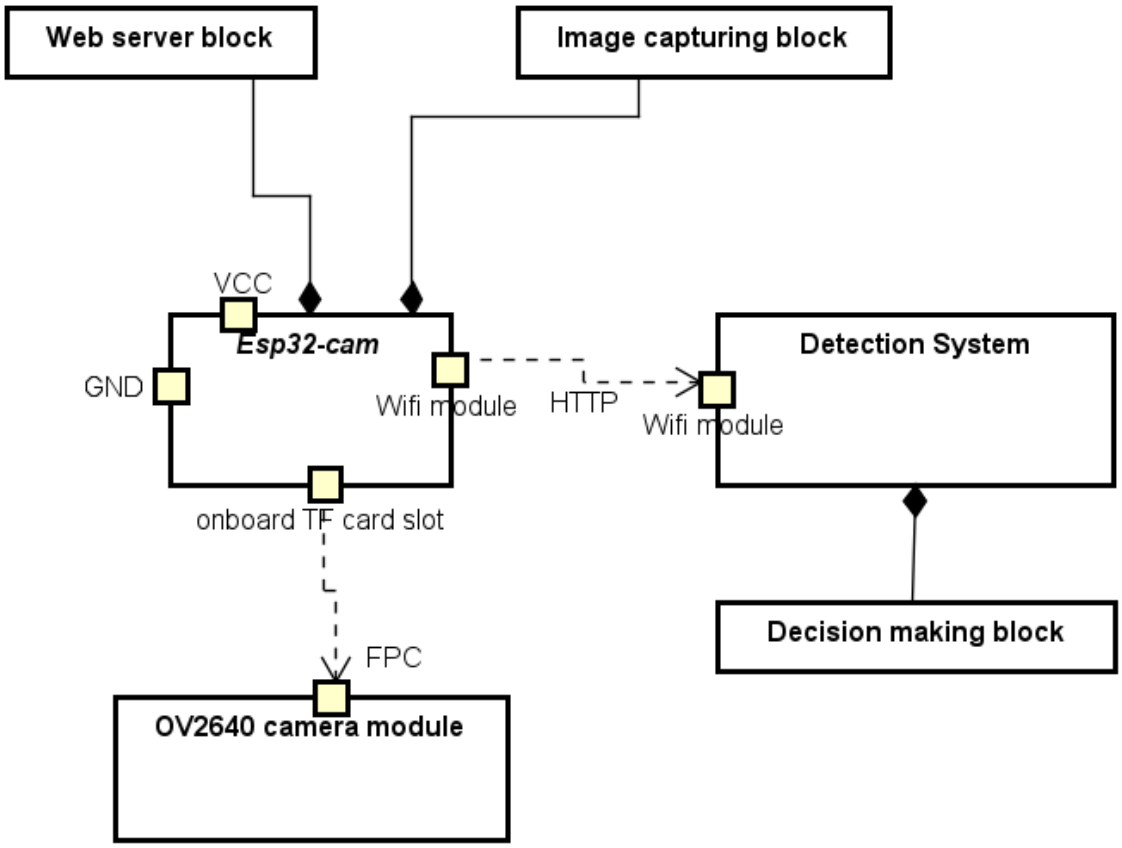



Figure 2.4: Block Definition Diagram version 0

# IV Project component

## 1 Hardware and Software environment

For a comprehensive understanding of the hardware and software environment components used in this study, I recommend referring to the annex chapter of this report. The annex contains detailed information on the system specifications and configurations, including hardware components such as ESP32-cam, as well as the software components such as application programs. By referring to the annex, you can gain a deeper insight into the technical details of the system, which can be helpful in evaluating the validity and reliability of the study results.

# V Workflow

We tried the approach of using contour detection to detect the logo on three different images. The first image was taken from a phone's 32mp camera, the second image was of the printed image of the phone's 32mp camera, captured using an OV2640 camera module, and the third image was of the actual physical product with an OV2640 camera module. However, before applying contour detection to these images, there were other necessary steps that needed to be taken

## 1 Brightness and contrast adjustment

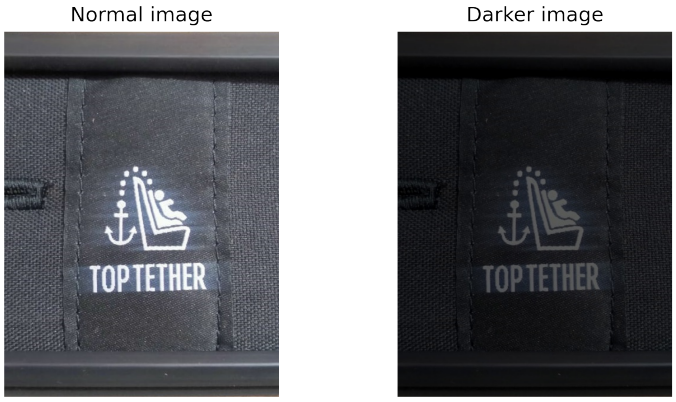



Figure 2.5: Phone's images : normal to darker

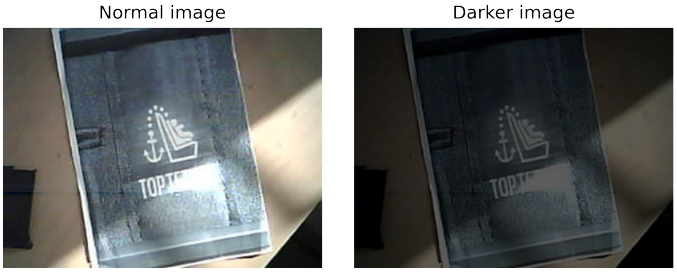



Figure 2.6: esp32-cam on printed images : normal to darker

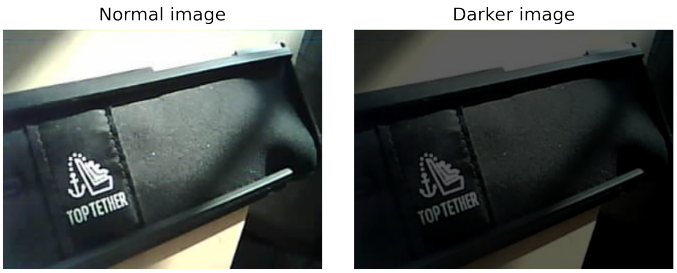



Figure 2.7: esp32-cam on the actual product : normal to darker

```
import cv2 as cv
import numpy as np

alpha = 0.4
beta = -10
result = cv.addWeighted(frame, alpha, np.zeros(frame.shape, frame.dtype), 0, beta)
```

The cv.addWeighted() function blends two input images by taking their pixel values and calculating a weighted sum for each corresponding pixel [11] using this equation:

$$output\_Pixel = \alpha * input1\_Pixel + \beta * input2\_pixel + \gamma[11] \tag{2.1}$$

where:

- output_Pixel = output pixel value
- $\alpha$ = weight given to the first input image (input1_pixel)
- input1_Pixel = first source array
- $\beta$ = weight given to the second input image (input2_pixel)
- input2_Pixel = second source array
- $\gamma$ = optional scalar value added to every output pixel value

## 2 reflection on the Brightness and contrast adjustment process

During my image processing experiments, I decided to test the effects of reducing brightness and adding contrast to images captured using different devices. To do this, I selected three images: the first was taken from a high-end 32 megapixel phone camera and it didn't have any problems with overexposure or loss of detail. The second image was a printed copy of the same photo, but this time using an ESP32-CAM as the capture device. Unfortunately, even with careful brightness reduction, a ray of sunlight was still visible in the image, and the overall quality was noticeably poorer than the first image. Finally, I used the ESP32-CAM to capture an image of the actual product in question, and again I observed a reduction in image quality compared to the high-end phone camera image. Even after applying the same brightness reduction and contrast enhancement techniques used on the other images, there was still visible noise and loss of detail, as well as a ray of sunlight that remained in the image. It seems that moving from a high-quality camera to an ESP32-CAM can have a significant impact on image quality, even with careful image processing techniques.

## 3 Grayscaling

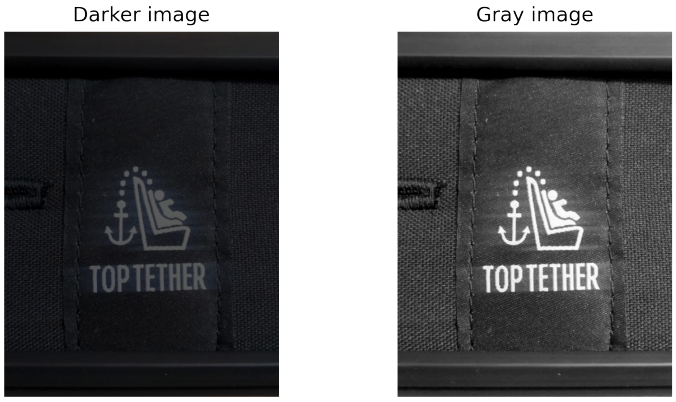



Figure 2.8: Phone's images :darker to gray

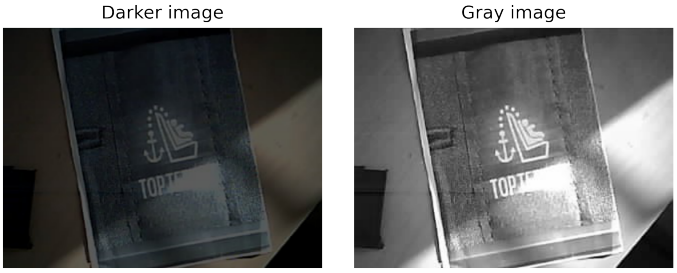



Figure 2.9: esp32-cam on printed images : darker to gray

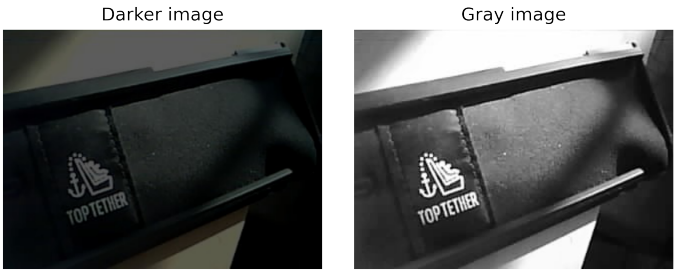



Figure 2.10: esp32-cam on the actual product : darker to gray

```
gray = cv.cvtColor(frame, cv.COLOR_BGR2GRAY)
```

The cv.cvtColor It takes as input the values of the red, green, and blue color channels of a pixel (R, G, B) and calculates a weighted average of these values to produce a single grayscale value using this equation:[11]

$$result = 0.299R + 0.587G + 0.114B[11] \tag{2.2}$$

Where:

- result: represents the grayscale value of a pixel in the output image.
- $R$: represents the red values of a pixel in the BGR image.
- $G$: represents the green values of a pixel in the BGR image.
- $B$: represents the blue values of a pixel in the BGR image.

## 4 Reflection on the grayscaling process

As part of my image processing experiments, I decided to apply grayscaling to the images as a preprocessing step before using contour detection to locate the logos. While this technique worked well for the first image, which was taken using a 32 megapixel phone camera and did not have any significant issues, it actually made the problems with the second and third images worse. Despite reducing the brightness and increasing the contrast, the ray of sunlight was still present in the images, and the grayscaling process made the problem more pronounced.

## 5 Image thresholding

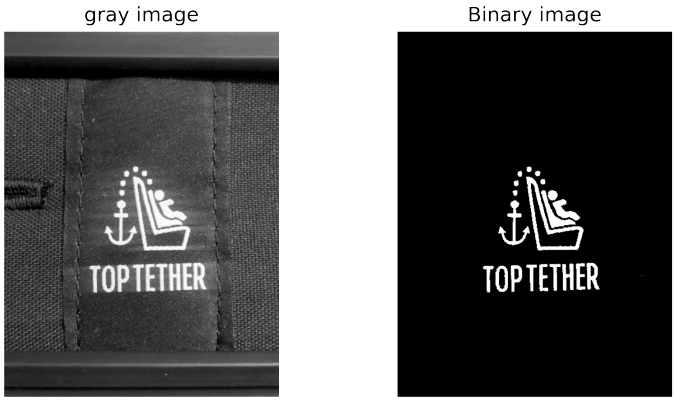



Figure 2.11: Phone's images :Gray to binary

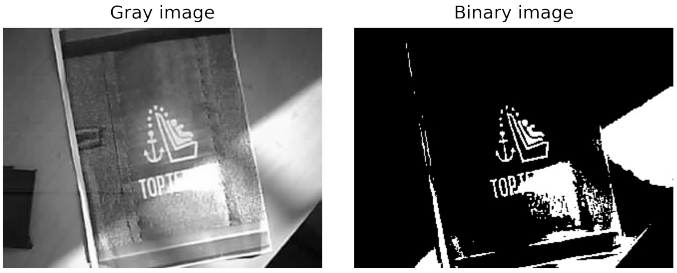



Figure 2.12: esp32-cam on printed images : Gray to binary

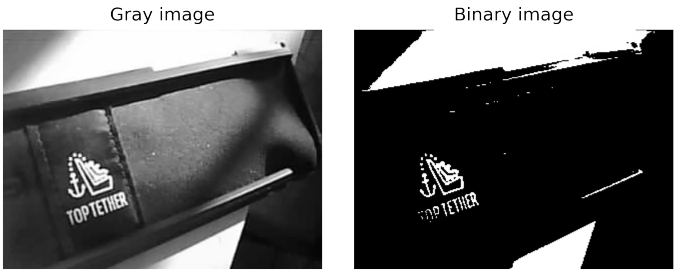



Figure 2.13: esp32-cam on the actual product :Gray to binary

```
ret, thresh = cv.threshold(adjusted, 70, 255, 0)
```

cv.threshold is used to threshold an image. It takes a gray image and applies a threshold to it [11] using this equation:

$$dst(x,y) = \begin{cases} maxVal & \text{if } src(x,y) > T \\ 0 & \text{otherwise} \end{cases} \text{ [11]} \tag{2.3}$$

- $dst(x,y)$: the output image pixel value at position $(x,y)$ after thresholding
- $maxval$: the maximum pixel value
- $src(x,y)$: the input image pixel value at position $(x,y)$
- $T$: the threshold value

## 6 Reflection on the image thresholding process

During the thresholding step, we didn't encounter any issues with the first image as the logo was clearly white and the background was black. However, the last two images posed a challenge as there was additional white noise caused by the light source, making it difficult to distinguish the logo from the noise.

## 7 Contour detection

In the contour detection step, we used a computer vision algorithm to identify and extract the logo from the images. While the algorithm worked perfectly fine for the first image captured from the 32 megapixel phone camera, it struggled to accurately detect the logo in the last two images taken with the ESP32 cam. The presence of the white noise in the images due to the sunlight made it difficult for the algorithm to distinguish between the logo and the background and oither wihite shapes, resulting in inaccurate or incomplete detection

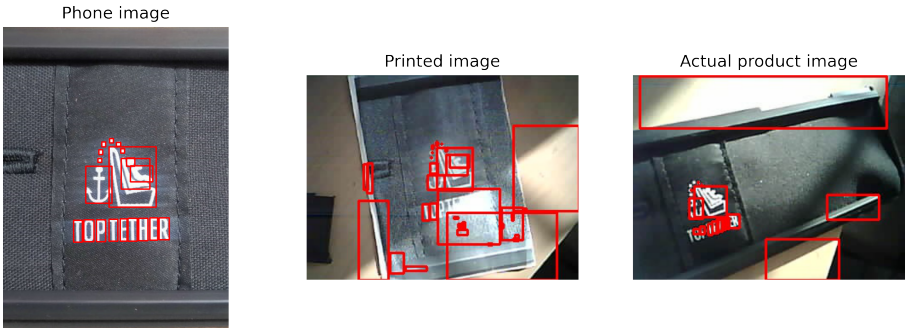



Figure 2.14: prediction results

```
contours, hierarchy = cv.findContours(thresh, cv.RETR_EXTERNAL, cv.CHAIN_APPROX_SIMPLE)
```

The `cv.findContours()` function in OpenCV is used to detect contours in a binary image. In our case, it uses the contour approximation method `CHAIN_APPROX_SIMPLE` [10], which is based on the Douglas-Peucker algorithm for curve simplification. This algorithm works as follows: we begin with a set of $n$ points $S = \{s_1, s_2, ..., s_n\}$ that represent a curve in two-dimensional space. We also have a tolerance value $\epsilon$. The algorithm proceeds as follows:

1. We calculate the line D that connects the first and last points in S.
2. We calculate the distance d(s, D) between each point s in S and the line D.
3. We then find the point smax in S that has the maximum distance d(smax, D).
4. If d(smax, D) is greater than $\epsilon$, we split the curve into two sub-curves at point smax, and we apply steps 1 to 3 recursively to each sub-curve.
5. Otherwise, we return the set of points s1, sn, smax.

To compute the distance $d(s, D)$ between a point $s$ and a line $D$, we use the following formula:

$$d(s, D) = \frac{|(s - s_1) \times (s_2 - p_1)|}{|s_2 - s_1|}$$

Here, $s_1$ and $s_2$ represent the two endpoints of the line $D$.

# VI Reflection on the contour detection process

We achieved successful results when applying contour detection to the phone image. The contour detection algorithm accurately detected every part of the logo, providing a precise boundary around it. This success can be attributed to several factors, such as the high resolution and clarity of the image, optimal lighting conditions, and well-defined edges of the logo. Overall, contour detection performed exceptionally well on the phone image.

In contrast to the phone image, we encountered challenges when applying contour detection to the printed image. The algorithm struggled to accurately detect the logo boundaries and produced several false positive detections. This difficulty can be attributed to various factors, such as variations in the printing quality, distortions introduced during the printing process, and noise present in the

image. These factors led to incomplete and inaccurate contour detection, resulting in the inclusion of unwanted objects or missing parts of the logo.

Similar to the printed image, the contour detection algorithm faced challenges when applied to the image of the actual product.

# VII Conclusion

In conclusion, the second chapter of our project focused on image processing techniques to extract the logo from images captured using the ESP32 cam. While we were able to successfully apply various techniques such as grayscaling, thresholding, and contour detection to extract the logo from the images, we faced significant challenges due to the presence of white noise in the images. Despite our best efforts, we were unable to achieve accurate results for all images. However, we learned valuable lessons and identified the limitations of traditional image processing techniques in solving this problem.

Moving forward, we plan to incorporate deep learning techniques in our next phase to improve the accuracy and efficiency of the logo detection system. Specifically, we will train a neural network to identify the logo in the images captured using the ESP32 cam, and we believe that this approach will provide better results than traditional image processing techniques. While this phase of our project did not produce the desired outcome, we remain optimistic and committed to finding a solution to this problem through continued experimentation and innovation.

# Chapter 3

# Raspberry pi 4 and ESP32 Cam Integration for Logo Detection

## I Introduction

This chapter is a step-by-step guide on how to train and use the YOLOv5 algorithm for logo detection. The process of logo detection involves identifying logos in images and videos, which can be challenging due to various factors such as logo occlusion, varying sizes and orientations, and the presence of similar objects in the background.

To address these challenges, the chapter walks through the various steps involved in building a logo detection model using the YOLOv5s algorithm, including data gathering and preprocessing, model training, validation, and testing.

This chapter coclude by diving into the steps needed to build a relay module for the communication between the raspberry pi and the automaton.

Overall, this chapter provides a comprehensive guide to building an accurate and efficient logo detection system using the YOLOv5s algorithm.

## II Specification of requirements

In this section, we introduce the different actors as well as the functional and non-functional requirements.

### 1 Description of functional requirements

- The ESP32-CAM board should be able to capture a live stream of pictures or videos and make them available through its built-in server.
- The raspberry pi should be able to connect to the server on the ESP32-CAM board and retrieve the images or videos.
- The raspberry pi should be able to process the images to determine if a specific logo is present or not.
- The raspberry pi should be able to process the images to determine if a specific logo is flipped on the x axis.
- The raspberry pi should be able to process the images to determine if a specific logo is in the first or the second half of the product.

- The raspberry pi should be able to provide some form of feedback or output to the automaton indicating the state of the logo.

## 2 Description of non-functional requirements

The requirements do not stop at the functional level but tend towards requirements that contribute to better quality of the application. The most important ones are:

- **Reliability:** The system should be able to consistently capture and process images accurately.
- **Performance:** The system should be able to process images quickly and without noticeable lag or delay.
- **Maintainability:** The code should be easy to maintain and update, with clear and well-documented code and configuration.

# III SysMl

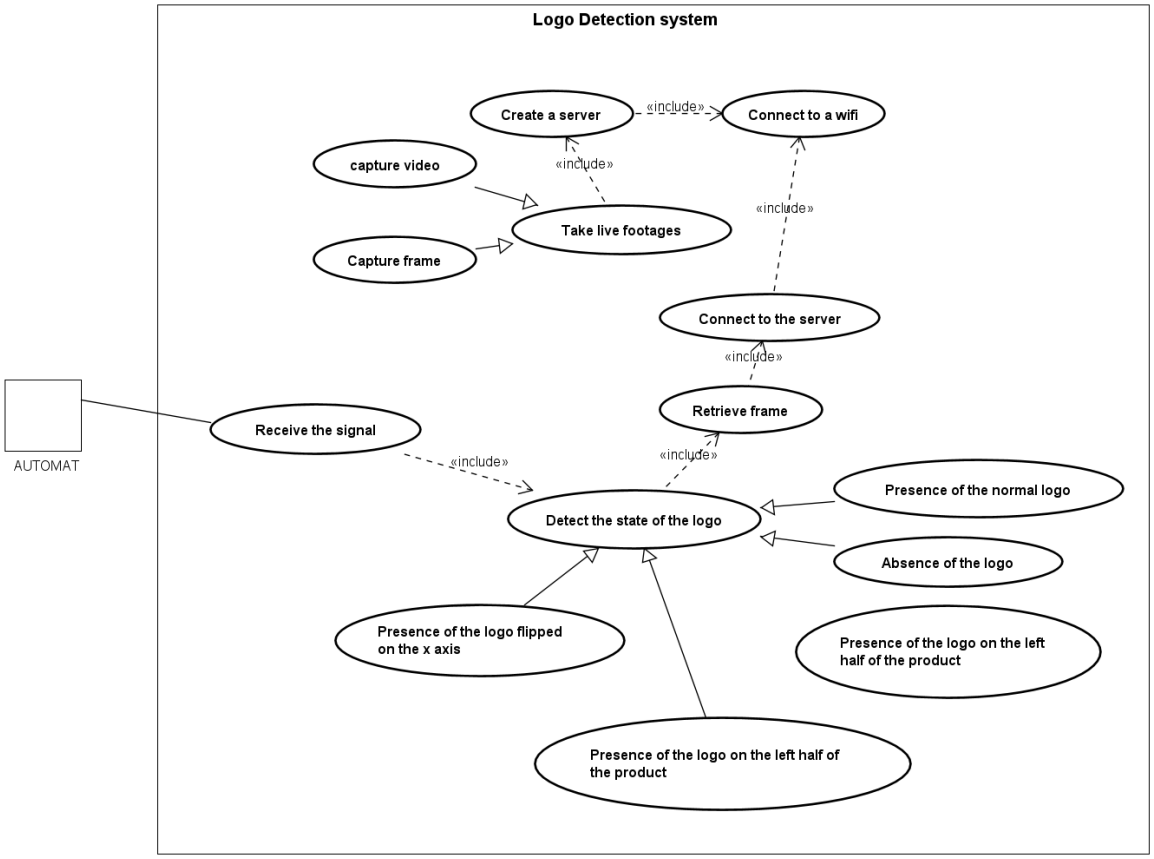



Figure 3.1: Use case diagram version 1

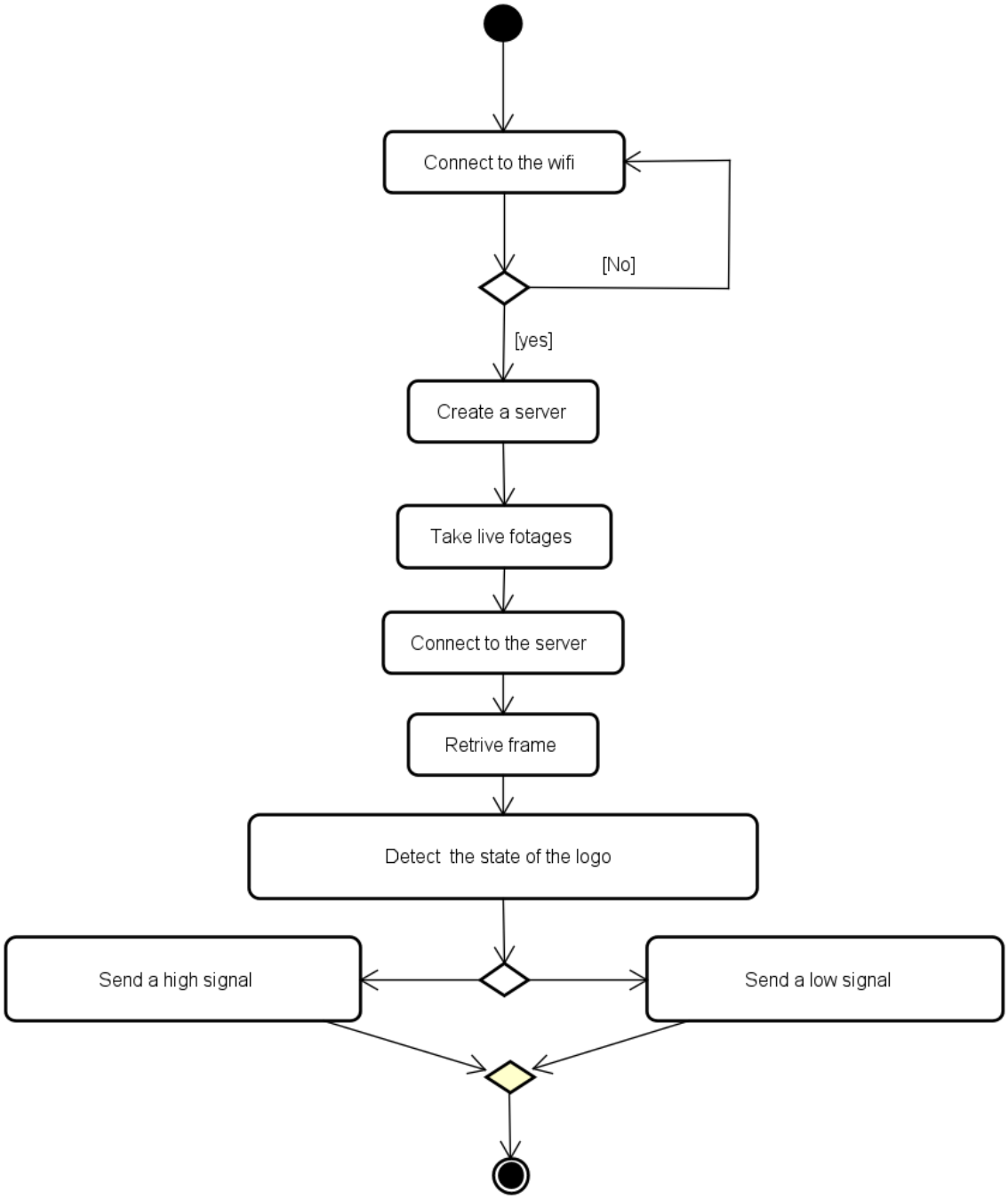



Figure 3.2: Activity diagram version 1

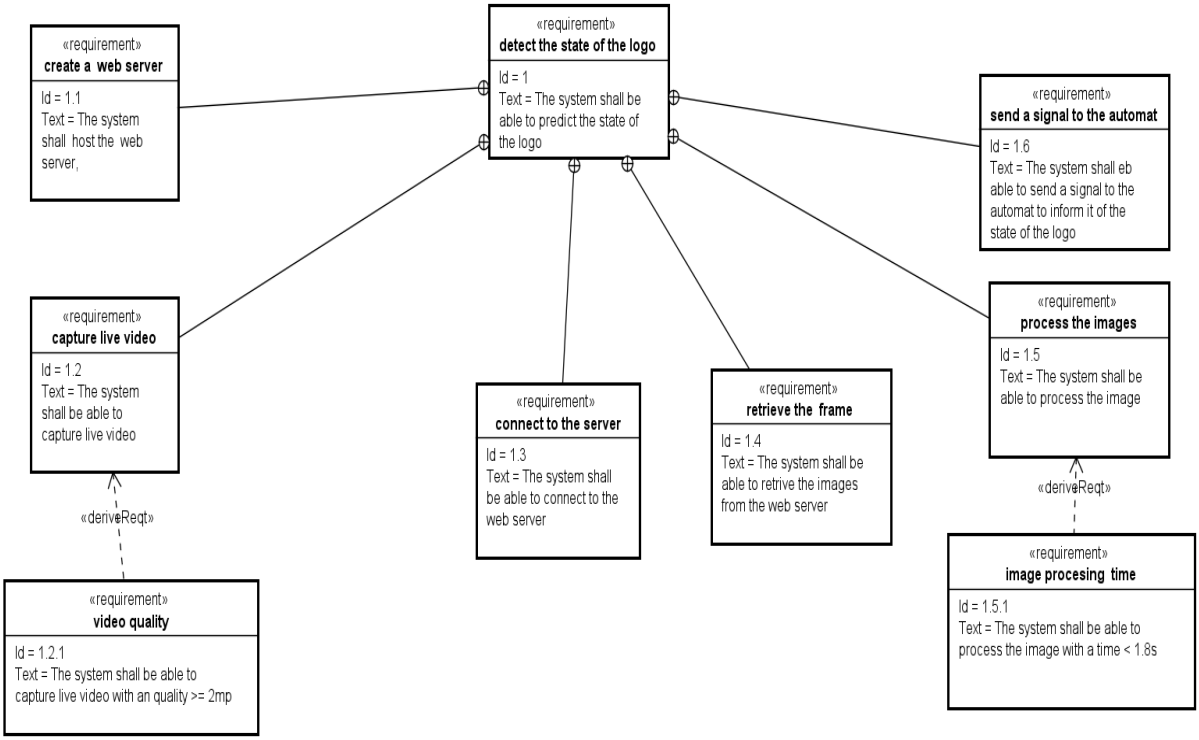



Figure 3.3: Requirement diagram version 1

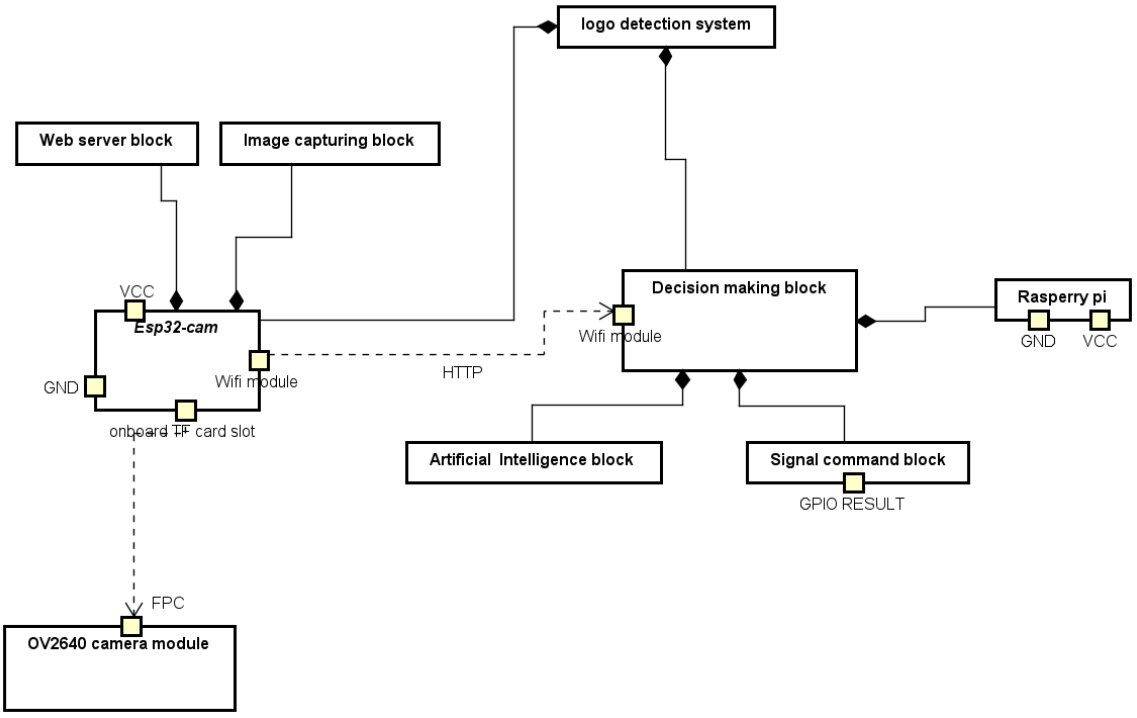



Figure 3.4: Block Definition Diagram version 1

# IV Project component

## 1 Hardware and Software environment

For a comprehensive understanding of the hardware and software environment components used in this study, I recommend referring to the annex chapter of this report. The annex contains detailed information on the system specifications and configurations, including hardware components such as ESP32-cam, as well as the software components such as application programs. By referring to the annex, you can gain a deeper insight into the technical details of the system, which can be helpful in evaluating the validity and reliability of the study results.

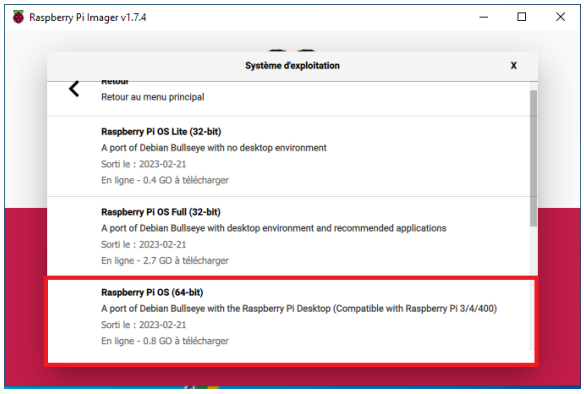
# V Workflow

## 1 Setting up the rapsberry pi

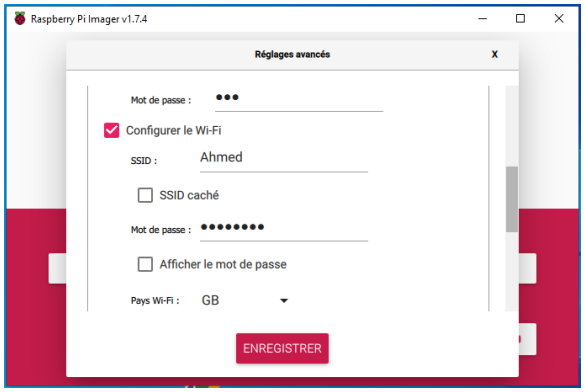
### 1.1 setting up the OS

- Download the Raspberry Pi Imager from the official Raspberry Pi website.

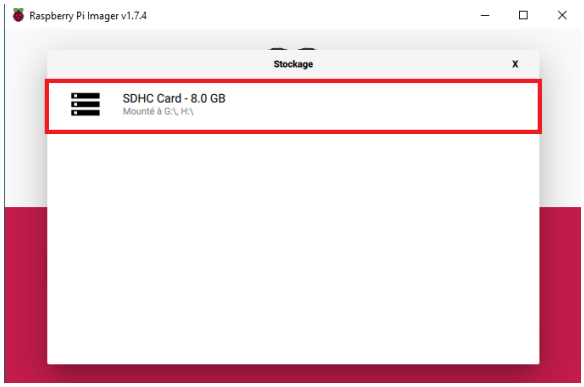

- Choose the raspberry pi operating system (64-bit)



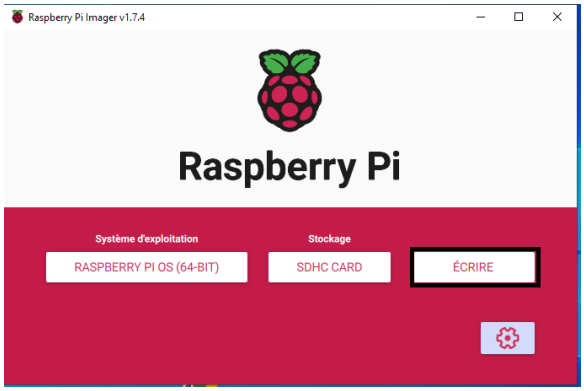

- Configure the parameters of the operating system: set a hostname,enable SSH,set username and password,configure wireless lan and set local settings



- Select the SD card you want to use for the installation.




- Click on the "Write" button to start the installation process. This will take several minutes to complete.




### 1.2 setting up the VNC

- Establishing connection with puttty

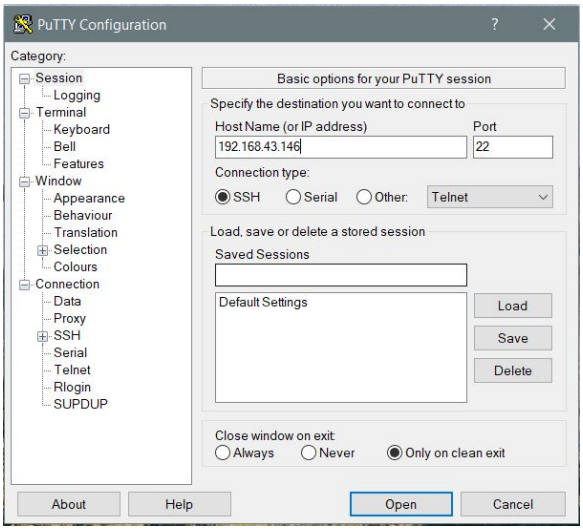



Figure 3.5: putty interface

- In the "Host Name (or IP address)" field, enter the IP address of your Raspberry Pi.
- In the "Port" field, enter "22". This is the default SSH port for Raspberry Pi.
- Under "Connection type", select "SSH".
- Click the "Open" button to start the SSH connection.

- Updating the system

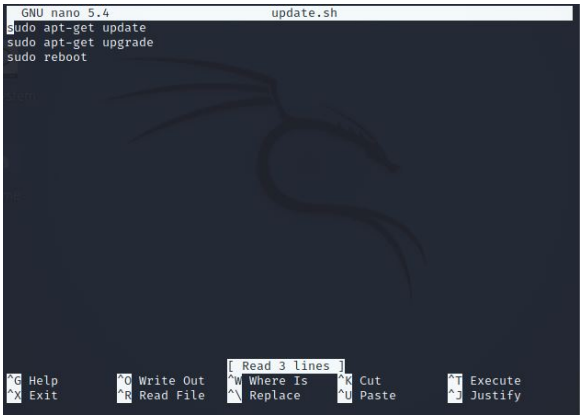



Figure 3.6: Update.sh file

I have created **update.sh** that contain these three commands:

- `> sudo apt-get update` : This command updates the list of available software packages and their versions from the repositories defined in the package manager sources list.
- `> sudo apt-get upgrade`: This command upgrades the installed packages on the system to their latest versions.
- `> sudo reboot`: This command restarts the system after the update and upgrade processes have completed.

To execute the update.sh just run this command: `> sh update.sh` This table represent the total time taking by this command:

| Execution time of sh update.sh | |
|---|---|
| **Metric** | **Value** |
| real | 9.631 s |
| user | 3.065 s |
| sys | 1.097 s |

- Activating VNC server

Enter this command: `> sudo raspi-config`

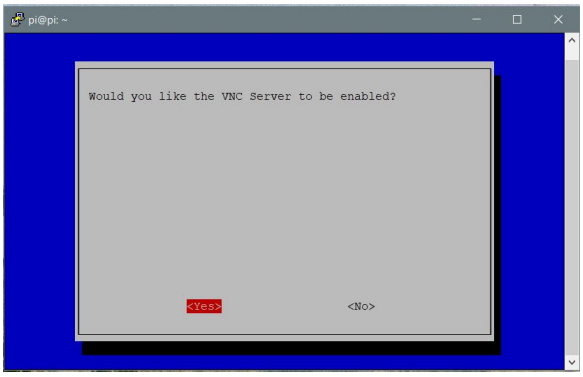



Figure 3.7: Enable vnc menu

Go to Interfacing options > vnc and click on “Select” While their, enable vnc

### 1.3 Prepare working environment

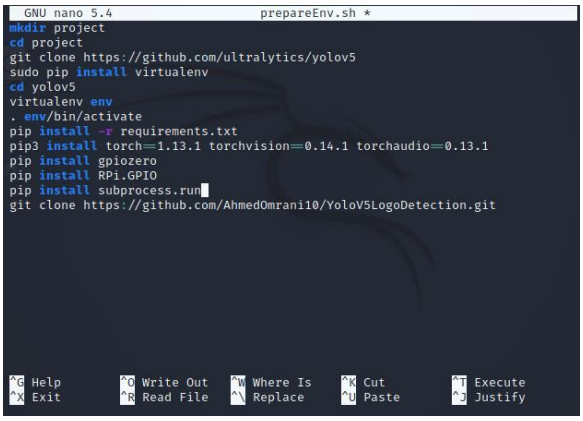



Figure 3.8: prepareEnv.sh file

I have created **prepareEnv.sh** that execute the following:

- `> mkdir project` :Create a new directory named `project` using the `> mkdir` command.
- `> cd project` :Change the current working directory to the `project` directory using the ”cd” command.
- `> git clone https://github.com/ultralytics/yolov5` :Clone the ”yolov5” repository from the GitHub account of ”ultralytics” using the `> git clone` command.
- `> sudo pip install virtualenv` :Install the ”virtualenv” package using the `> sudo pip install` command, which creates a virtual environment in the `env` directory.
- `> cd yolov5` :Change the current working directory to the `yolov5`directory using the `> cd` command.

- `> . env/bin/activate` :Activate the virtual environment created earlier using the `> .` command followed by the path to the `env/bin/activate` file.
- `> pip install -r requirements.txt` :Install the Python packages listed in the "requirements.txt" file using the `> pip install -r` command.
- `> pip3 install torch==1.13.1 torchvision==0.14.1 torchaudio==0.13.1` :Install torch version 1.13.1 torchvision version 0.14.1 torchaudio version 0.13.1 packages using the `> pip3 install` command because the the ones in "requirements.txt" file are not compatible with the current yolov5.
- `> pip install gpiozero` :Install the `gpiozero` package using the `> pip install` command.
- `> pip install RPi.GPIO` :Install the `RPi.GPIO` package using the `> pip install` command.
- `> pip install subprocess.run` :Install the `> subprocess.run` package using the `> pip install` command.
- `> git clone https://github.com/AhmedOmrani10/YoloV5LogoDetection.git` :Clone the "YoloV5LogoDetection" repository from the GitHub account of "AhmedOmrani10" using the `> git clone` command.

To execute the prepareEnv.sh just run this command: `> sh prepareEnv.sh` This table represent the total time taking by the command:

| Execution time of prepareEnv.sh | |
|---|---|
| **Metric** | **Value** |
| real | 18.759 m |
| user | 3.334 s |
| sys | 31.253 s |

## 2 Deep learning pipeline

- Selecting the deep learning model: Why YOLOv5s?// Before delving into the reasons for choosing YOLOv5s, it is important to understand some of the key metrics used to evaluate object detection models.
  - Precision:

$$P = \frac{TP}{TP + FP} \tag{3.1}$$

Precision, also known as positive predictive value, is a metric used in machine learning to evaluate the accuracy of a classification model. It is calculated as the ratio of true positives to the sum of true positives and false positives. When expressed as a probability, it represents the likelihood that a randomly selected instance classified as positive is actually a true positive. A perfect classifier with no false positives has a precision of 1.[20]

– Recall:

$$R = \frac{TP}{TP + FN} \tag{3.2}$$

Recall, also known as sensitivity, is a performance metric used in machine learning that measures the proportion of positive instances that are correctly identified as positive by the model. When expressed as a probability, it represents the likelihood that a randomly selected true positive instance will be correctly classified as positive by the model. Unlike precision, which only takes into account instances predicted as positive by the model, recall considers all actual positive instances.[20]

– mAP:

$$mAP = \frac{1}{n} \sum_{k=1}^{N} AP_k \tag{3.3}$$

mAP is a metric based on the area under precision-recall curve (PRC) that is preprocessed to eliminate zig-zag behavior (Padilla et al., 2021 [1]). Where $AP_k$ is the average precision (AP) of class $k$ and $n$ is the number of thresholds. The mAP values were calculated at $Z$ value threshold value for intersection over union (IoU) meaning, all the predicted bounding boxes that resulted in ratios of overlapping areas to the union areas with ground truth bounding greater than $Z$ were considered and the remaining were discarded.[21]

– IoU:

$$IoU(A, B) = \frac{A \cap B}{A \cup B}, \quad IoU(A, B) \in [0, 1] \tag{3.4}$$

Intersection over Union (IoU) is a commonly used method to evaluate object detection algorithms. It measures the overlap between the proposed bounding box and the ground truth bounding box by computing the ratio of their intersection over their union. If the resulting value is above a certain threshold, the proposed bounding box is considered a correct detection.[21]

In the two tables below, we compare the performance of different YOLOv5 models.

**Comparison of YOLOv5 Models [22]**

| Model | Size (pixels) | mAPval@50-95 | mAPval@50 |
|---|---|---|---|
| YOLOv5n | 640 | 28.0 | 45.7 |
| YOLOv5s | 640 | 37.4 | 56.8 |
| YOLOv5m | 640 | 45.4 | 64.1 |
| YOLOv5l | 640 | 49.0 | 67.3 |
| YOLOv5x | 640 | 50.7 | 68.9 |

All YOLOv5 checkpoints in the table are trained up to 300 epochs with default settings. The Nano and Small models use hyp.scratch-low.yaml hyps, while all other models use hyp.scratch-high.yaml. The mAPval values shown in the table are for single-model single-scale on the COCO val2017 dataset. To reproduce these results, one can run the command "python val.py –data coco.yaml –img 640 –conf 0.001 –iou 0.65".[22]
The speed measurements shown in the table are averaged over COCO val images using an AWS p3.2xlarge instance. The NMS times, which take about 1 ms per image, are not included in these measurements. To reproduce these results, one can run the command "python val.py –data coco.yaml –img 640 –task speed –batch 1".[22]
The TTA (Test Time Augmentation) Test Time Augmentation includes reflection and scale augmentations. To reproduce these results, one can run the command "python val.py –data coco.yaml –img 1536 –iou 0.7 –augment".[22]

| Comparison of YOLOv5 Models part2 [22] | | | | |
|---|---|---|---|---|
| **Model** | **Speed CPU b1 (ms)** | **Speed V100 b1 (ms)** | **Speed V100 b32 (ms)** | **Params (M)** |
| YOLOv5n | 45 | 6.3 | 0.6 | 1.9 |
| YOLOv5s | 98 | 6.4 | 0.9 | 7.2 |
| YOLOv5m | 224 | 8.2 | 1.7 | 21.2 |
| YOLOv5l | 430 | 10.1 | 2.7 | 46.5 |
| YOLOv5x | 766 | 12.1 | 4.8 | 86.7 |

All YOLOv5 checkpoints in the table are trained up to 300 epochs with default settings. The Nano and Small models use hyp.scratch-low.yaml hyps, while all other models use hyp.scratch-high.yaml. The mAPval values shown in the table are for single-model single-scale on the COCO val2017 dataset. To reproduce these results, one can run the command "python val.py –data coco.yaml –img 640 –conf 0.001 –iou 0.65".[22]
The speed measurements shown in the table are averaged over COCO val images using an AWS p3.2xlarge instance. The NMS times, which take about 1 ms per image, are not included in these measurements. To reproduce these results, one can run the command "python val.py –data coco.yaml –img 640 –task speed –batch 1".[22]
The TTA (Test Time Augmentation) Test Time Augmentation includes reflection and scale augmentations. To reproduce these results, one can run the command "python val.py –data coco.yaml –img 1536 –iou 0.7 –augment".[22]

The **YOLOv5** model is a state-of-the-art object detection model that has gained popularity due to its high accuracy and speed. The YOLOv5 family includes several models such as YOLOv5n, YOLOv5s, YOLOv5m, YOLOv5l, and YOLOv5x, each with different sizes and performance characteristics. To determine the best model for our application of logo detection on a Raspberry Pi 4, we used those table above to compare the YOLOv5 models using metrics such as mAPval@50-95, mAPval@50, speed on CPU and V100, number of parameters, and FLOPs.

Firstly, let's consider the Yolov5n model, which has the smallest size(1.9 million params) and computational requirements in the Yolov5 family. While this model is suitable for low-resource devices, it sacrifices accuracy (mAPval50-95 = 28.0 and mAPval50 = 45.7) for speed ( CPU b1 = 45 ms , Speed V100 b1 = 6.3 ms and Speed V100 b32 = 0.6 ms), making it less suitable for complex object detection tasks.

Next, the Yolov5m model strikes a balance between accuracy(mAPval50-95 = 45.4 and mAPval50 = 64.1) and speed ( CPU b1 = 224 ms, Speed V100 b1 = 8.2 ms and Speed V100 b32 = 1.7 ms), making it a popular choice for most applications. However, it still requires significant computational power (21.2 million params), which may not be feasible for low-resource devices like the Raspberry Pi 4.

The Yolov5l and x models offer the highest accuracy (mAPval50-95 = 49.0 and mAPval50 = 67.3 and mAPval50-95 = 50.7 and mAPval50 = 68.9 ) but come with even greater computational requirements (46.5 and 86.7 million params). They are best suited for high-end GPUs or cloud computing platforms and are not ideal for the Raspberry Pi 4.

Finally, we come to Yolov5s, which is the best choice for the Raspberry Pi 4. It has a smaller size and computational requirements (7.2 million params) than the other models in the family, making it easier to deploy on low-resource devices. Additionally, Yolov5s achieves high accuracy (mAPval50-95 = 37.4 and mAPval50 = 56.8) while maintaining a fast inference speed ( CPU b1 = 98 ms, Speed V100 b1 = 6.4 ms and Speed V100 b32 = 0.9 ms), making it an excellent choice for object detection tasks that require real-time performance.

In summary, the Yolov5s model is the best choice for the Raspberry Pi 4 due to its small size, low computational requirements, and high accuracy. It strikes the perfect balance between speed and accuracy, making it ideal for most object detection tasks.

- Building our dataset

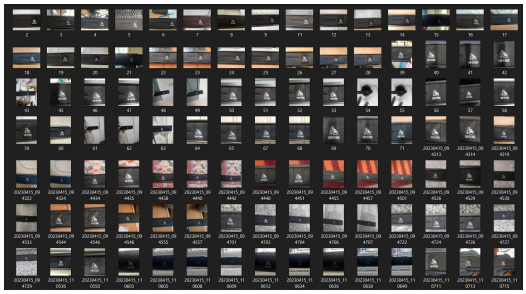


Figure 3.9: Normal dataset

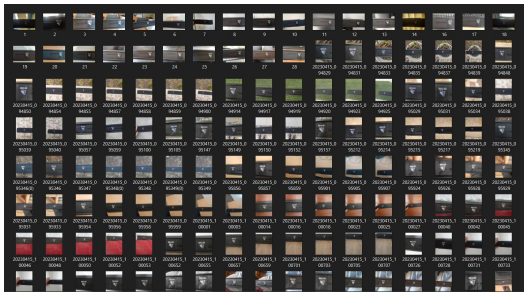


Figure 3.10: Inverted dataset

To train our Yolov5s model, we first gathered a dataset of 2500 images containing logos. We split the dataset into two classes: Normal and Inverted. The Normal class 4.5 contained 1250 images of logos in their standard orientation, while the Inverted class 4.6 contained 1250 images of logos that had been rotated 180 degrees.

- Image labeling

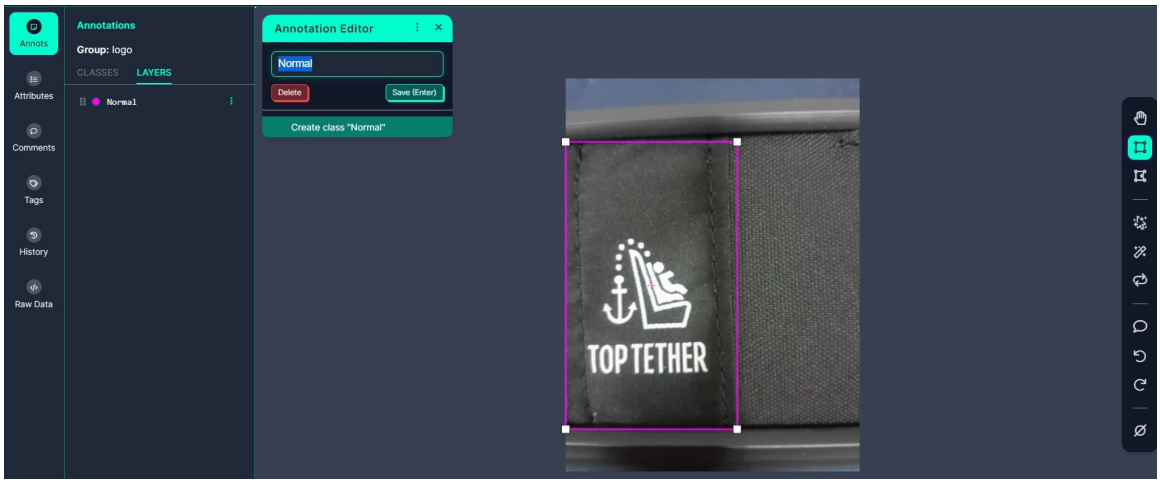



Figure 3.11: Labeling an image

I used Roboflow to perform the image labeling. Image labeling involves annotating the images with bounding boxes that define the location of the logos in each image. This step is essential for training an object detection model like Yolov5s. After hours of scouring through an endless stream of images and yelling at my computer screen, I finally finished the task of labeling the images for our dataset. With every annotation made, I felt as if a weight had been lifted from my shoulders (and added to my mouse finger). The satisfaction of seeing all of our hard work come together was almost as great as the relief of knowing I won't have to do it again anytime soon. Now, with our dataset fully labeled and ready for training, it's time to sit back, relax, and let the computer do the heavy lifting.

- Image data augmentation

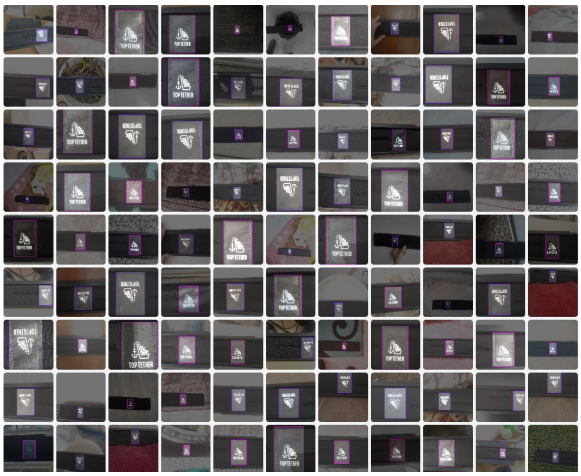


Figure 3.12: Augmented dataset

For data augmentation, we used several techniques in Roboflow, including Hue, Saturation, Brightness, and Blur. Hue refers to the color of the image, and we applied random rotations between -11° and +11° to create variations in color. Saturation refers to the intensity of the colors, and we applied random variations between -14% and +14% to increase the diversity of the dataset. Brightness refers to the overall brightness of the image, and we applied random variations between 0% and +36% to simulate different lighting conditions. Finally, we applied a blur of up to 0.75 pixels to simulate blurring caused by motion or other factors.

After augmenting our dataset, we ended up with triple the number of images, which allowed us to train a more robust model.

- Data loading

```
# Import the Roboflow package
from roboflow import Roboflow

# Instantiate a Roboflow object and provide your API key
rf = Roboflow(api_key="AKSDKS85skldkMLD755S")

# Specify the project and dataset that you want to download
project = rf.workspace("ahmed-omrani-uqhfw").project("logodetection-fmddc")
dataset = project.version(3).download("yolov5")
```

The training data was downloaded using the Roboflow Python package, which is a popular tool for managing and preprocessing computer vision datasets. The training data was downloaded from a specific Roboflow project and was in the YOLOv5 PyTorch format

- Transfer learining on yolov5s

Fine-tuning is a popular technique in machine learning that involves taking a pre-trained model and adapting it to a new problem. Transfer learning is the process of using a pre-trained model to jumpstart a new model's training. These techniques are widely used in computer vision tasks where labeled data is limited, and collecting more data is costly. By fine-tuning a pre-trained model, the model can leverage its knowledge and adapt to the new task quickly. The process typically involves freezing some of the pre-trained layers and only updating the weights of the

output layer or a few top layers. By doing so, the model can adapt its weights to the new problem while retaining its knowledge of the previously learned features. This approach often leads to improved performance with fewer labeled examples.[28]

In our case, the YOLOv5s model was fine-tuned to detect two classes: normal logos and inverted logos, instead of the original model which was trained on 80 classes. The model was modified to recognize only these two specific classes by adjusting the number of classes parameter in the YOLOv5s model's configuration file. The configuration file, which is a YAML file that specifies the network architecture and its parameters, was updated to reflect the change in the number of classes. This customization allows the model to be optimized for the specific task of detecting normal and inverted logos, rather than having to account for the nuances of 80 different classes. By using transfer learning, only the last layer of the YOLOv5s model was modified to detect the two new classes. The other layers of the model were left unchanged, as they were already trained to detect objects. This means that the feature extraction layers of the model are kept intact and only the classification layers are modified. The modified configuration file was saved as custom_yolov5s.yaml. This file was used to train the fine-tuned YOLOv5s model.

- Training the fine-tuned yolovs

  Once the data was downloaded, the training process was initiate. The model was trained using the train.py script from the YOLOv5 repository. The script was configured to use the custom configuration file and the downloaded training data. To train the YOLOv5 model, we use the following line of code:

```
!python train.py --img 416 --batch 16 --epochs 300 --data
    {dataset.location}/data.yaml --cfg ./models/custom_yolov5s.yaml --name
    yolov5s_results --cache
```

  During the training process, the weights of the YOLOv5s model were updated to detect only two classes. The model was trained for 300 epochs, with a batch size of 16 and an image size of 416x416. The model was also cached, which means that the weights of the model were saved after each epoch, reducing the training time in the future.

| Training Metrics for YOLOv5s | | | |
|---|---|---|---|
| **Precision** | **Recall** | **mAP50** | **mAP50-95** |
| 0.998 | 1.000 | 0.995 | 0.889 |

  The training output shows that the model achieves a precision of 0.999, a recall of 1, mAP50 of 0.995, and mAP50-95 of 0.889. These metrics indicate that the model has learned to detect objects with high accuracy.

- Validating the fine-tuned yolovs Validation is a critical step in the training process, which involves evaluating the performance of the trained model on a validation dataset. The validation dataset is a separate dataset from the training dataset that is used to ensure that the model is not overfitting to the training data.

  To validate the YOLOv5 model, we use the following line of code :

```
!python val.py --weight "runs/train/yolov5s_results/weights/best.pt" --data
    "/content/yolov5/LogoDetection-3/data.yaml"
```

The val.py script loads the trained model and evaluates it on the validation dataset specified in the data.yaml file. The –weight flag specifies the path to the weights of the trained model, and the –data flag specifies the path to the dataset configuration file.

| Validation Metrics for YOLOv5s | | | |
|---|---|---|---|
| **Precision** | **Recall** | **mAP50** | **mAP50-95** |
| 0.999 | 1.000 | 0.955 | 0.877 |

The validation output shows that the model achieves a precision of 0.99, a recall of 1, mAP50 of 0.955, and mAP50-95 of 0.877. These metrics indicate that the model is performing well on the validation dataset and didn't overfit that much to the training data.

- Testing the fine-tuned yolovs Testing is the process of evaluating the performance of the trained model on a completely new dataset that the model has never seen before. It is used to ensure that the model can generalize well to new data. To test the YOLOv5 model, we use the following line of code:

```
!python val.py --weight "runs/train/yolov5s_results/weights/best.pt" --data
    "/content/yolov5/LogoDetection-3/data.yaml" --task test
```

This command is used to evaluate the performance of the trained YOLOv5 model on a test dataset. The val.py script is used here again, but this time with the –task test flag which tells the script to evaluate the model on the test dataset. The –weight flag specifies the path to the best weights file obtained during training. The –data flag specifies the path to the dataset YAML file containing information about the test dataset.

| Testing Metrics for YOLOv5s | | | |
|---|---|---|---|
| **Precision** | **Recall** | **mAP50** | **mAP50-95** |
| 0.999 | 0.997 | 0.954 | 0.878 |

The test output shows that the model achieves a precision of 0.999, a recall of 0.977, mAP50 of 0.994, and mAP50-95 of 0.878. These metrics indicate that the model is performing well on the test and didn't overfit that much to the training data.

- Classes Detection

After completing the crucial steps of image gathering, image labeling, image augmentation, and training the YOLOv5 model, we finally reach the intense step of inference. Inference is the process of using the trained model to make predictions on new, unseen images. This step is the culmination of all the hard work put into building and refining the model, and it is where the model's true effectiveness is put to the real-life test. Through inference, we can evaluate the model's accuracy and see how well it performs on real-world data, ultimately determining its usefulness in solving the problem at hand.[8]
During inference, the YOLOv5 algorithm takes an input image and passes it through a convolutional neural network (CNN) to extract feature maps. The CNN uses several layers of convolution and pooling operations to extract features from the image. These features are then passed through a series of fully connected layers to produce a set of bounding boxes and associated confidence scores for each detected object.

The bounding boxes are defined by their coordinates (x, y) and their width and height (w, h). The confidence scores indicate the probability that an object is present in the bounding box.

During inference, the YOLOv5 algorithm evaluates the confidence scores for each detected object and selects the object with the highest score as the primary detection.


Figure 3.13: Normal class detection


Figure 3.14: Inverted class detection

Figure 3.15: Detection of nothing

As we can see in figure 4.6 If an object is in an inverted position, the YOLOv5 algorithm will still be able to detect it. This is because the CNN is able to learn and recognize patterns and features of objects, regardless of their orientation in the image. The algorithm will simply detect the object and output its bounding box and confidence score, regardless of whether it is in a normal in figure 4.6 or inverted position in figure 4.5 and if it doesn't we can conclude that there is no logo in figure 4.7.


Figure 3.16: Logo on the left side detection

Now To detect if the logo is printed on the right or left side of the product, you can use image processing techniques. First, you can extract the coordinates of the bounding box resulting from the YOLOv5 prediction of the normal class as shown in the figure 4.8 by assigning xmin and xmax to the variables row[0] and row[2] respectively.

```
1 xmin, xmax = row[0],row[2]
```

Next, you can calculate the center of the bounding box by finding the average of xmin and xmax. This can be done with the following code:

```
center_x = (xmin + xmax) / 2
```

Then, you can calculate the width of the frame using the shape attribute of the frame, which returns the height, width, and number of channels of the image. This can be done with the following code:

```
width = frame.shape[1]
```

The code above calculates the width of the image using the shape attribute of the frame, but only assigns the second element (index 1) to the variable width, which corresponds to the width of the image. The height and number of channels are not used in this code. Finally, you can compare the center of the bounding box with the center of the image. If the center of the bounding box is smaller than the center of the image, it means that the logo is located on the left side of the product, which is a defective piece. If it is on the right side, it means that the logo is printed on the right side of the product, which is a normal piece.

# VI Experimental Setup and Results

To determine which Raspberry Pi model is better suited for running the custom YOLOv5s model, we conducted an experiment where we performed 500 object detections on both Raspberry Pi 4 and Raspberry Pi 3. The goal was to compare their performance in terms of detection time.

## 1 Experimental Procedure

The following steps were followed during the experiment:

1. The custom YOLOv5s model was deployed on both Raspberry Pi 4 and Raspberry Pi 3.
2. A test dataset consisting of 500 images was prepared for object detection.
3. The model ran on each image in the dataset, and the time taken for detection was recorded.
4. The number of detections, average, minimum, and maximum detection time were calculated for each Raspberry Pi model.

## 2 Results

The results of the experiment are summarized in Table below:

**Detection time results for Raspberry Pi 4 and Raspberry Pi 3**

| Raspberry Pi Model | Number of Detections | Average Detection Time (s) | Range (s) |
|---|---|---|---|
| Raspberry Pi 4 | 500 | 0.8843 | 0.8687 - 0.9248 |
| Raspberry Pi 3 | 500 | 3.2731 | 3.2186 - 3.3478 |

## 3 Discussion

From the results, it is evident that Raspberry Pi 4 outperformed Raspberry Pi 3 in terms of detection time. The average detection time on Raspberry Pi 4 was approximately 0.8843 seconds, whereas on Raspberry Pi 3, it was around 3.2731 seconds. This significant difference indicates that Raspberry Pi 4 is better suited for running the custom YOLOv5s model.

These findings highlight the importance of hardware capabilities when it comes to real-time object detection tasks. Raspberry Pi 4's improved processing power and memory capacity contribute to its superior performance in this scenario.

Overall, based on the experiment results, we can conclude that Raspberry Pi 4 is the recommended choice for running the custom YOLOv5s model, as it provides faster and more efficient object detection.

## 4 Relay module desgin

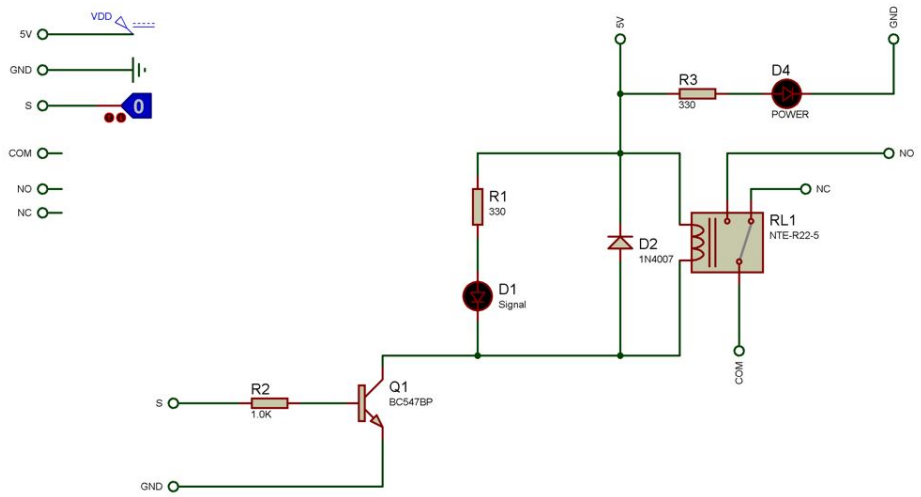



Figure 3.17: Schematic Capture of the relay

When interfacing a Raspberry Pi, which operates at 3.3V logic levels, with a device that requires a higher voltage, such as 24V in my case, using a relay becomes essential. The 3.3V GPIO signal from the Raspberry Pi may not be strong enough to directly trigger a response from the target device, like the automaton in your setup.

A relay acts as an electrical switch that can control higher voltages and currents using a lower voltage control signal. In this scenario, the 3.3V signal from the Raspberry Pi activates the 5V relay's coil, which then switches its contacts to allow the passage of the higher voltage signal (in this case, 24V) to the automaton.

By utilizing the relay, you create an isolation barrier between the low voltage control circuit (Raspberry Pi) and the high voltage load circuit (automaton). This isolation protects the Raspberry

Pi's delicate electronics from potential damage caused by the higher voltage, ensuring safe and reliable operation.

In my project, I designed a PCB relay module using Proteus software. I started by creating a new project and selecting the schematic capture option.As you can see in figure 3.18, I added all the necessary components such as the relay, terminal blocks, and supporting elements like diodes or resistors. I carefully connected the components together, making sure the signals flowed correctly by drawing wires between their pins. To ensure clarity, I included labels and annotations.

Once the schematic was complete, I performed a design rule check to catch any errors or violations. I made the necessary revisions to ensure the schematic was error-free and ready for the next step.

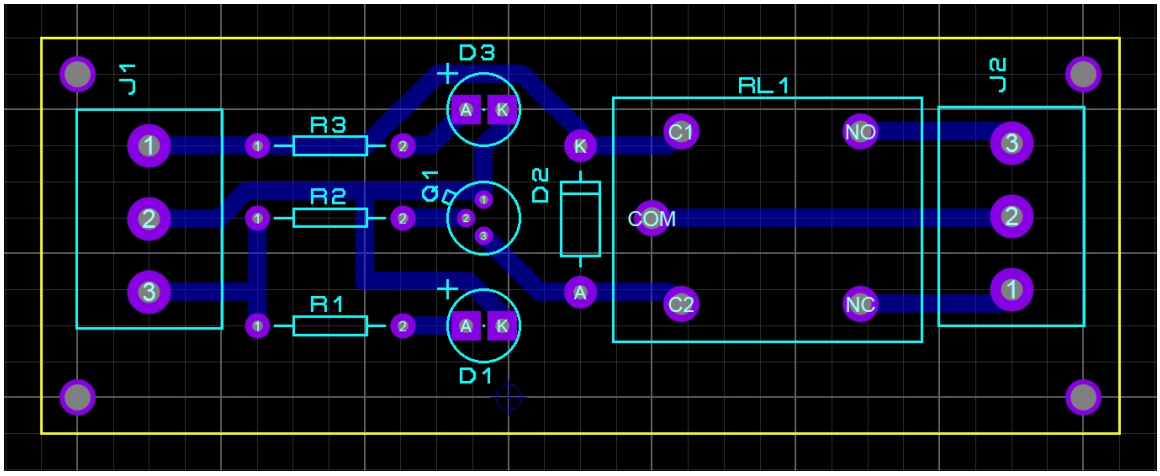



Figure 3.18: Schematic Capture of the relay

Moving on to the PCB layout stage, I selected the PCB layout option in Proteus and imported the schematic. This generated an initial component placement, which I organized and optimized for efficient routing. I took into consideration factors like signal paths, component clearance, and the overall size of the PCB.

With the component placement finalized, I began routing the connections using Proteus's routing tools. I created traces that connected the appropriate pins of the components, following design rules and guidelines. I paid extra attention to critical paths and sensitive signals, ensuring proper separation and avoiding noise issues.

# VII Conclusion

In conclusion, implementing YOLOv5 proved superior to contour detection in detecting the four different states of the logo: normal, inverted, absent, and on the left side. However, new challenges emerged that need to be addressed in the next chapter such as new defective states of the logo , network improvements and more.

# Chapter 4

# Stabilizing the System

## I Introduction

In this chapter, we made several significant changes to our project that have improved its functionality and usability. First, we decided to switch from using a traditional router to using a Raspberry Pi as an access point. We also made the decision to give our ESP32-CAM server a static IP address.

In addition, we identified an issue with the position of the logo and the font. To address this, we used image processing with the addition of a new class to the YOLOV5s to help us accurately identify and classify these types of objects.

Furthermore, I am excited to announce that I automated the main script for inference and I fixed the raspberry pi's temperature issue.

## II Specification of requirements

In this section, we introduce the different actors as well as the functional and non-functional requirements.

### 1 Description of functional requirements

- The system should be able to capture a live stream of pictures or videos and make them available through its built-in server.
- The system should be able to connect to the server on the ESP32-CAM board and retrieve the images or videos.
- The system should be able to process the images to determine if a specific logo is present or not.
- The system should be able to process the images to determine if a specific logo is flipped on the x axis.
- The system should be able to process the images to determine if a specific logo is in the first or the second half of the product.
- The system should be able to process the images to determine if a specific the font of the logo have a problem.
- The system should be able to process the images to determine if a specific the logo is under the normal postion.

- The system should be able to provide some form of feedback or output to the automaton indicating the state of the logo.

## 2 Description of non-functional requirements

The requirements do not stop at the functional level but tend towards requirements that contribute to better quality of the application. The most important ones are:

- **Reliability:** The system should be able to consistently capture and process images accurately..
- **Performance:** The system should be able to process images quickly and without noticeable lag or delay.
- **Maintainability:** The code should be easy to maintain and update, with clear and well-documented code and configuration.

# III SysMl

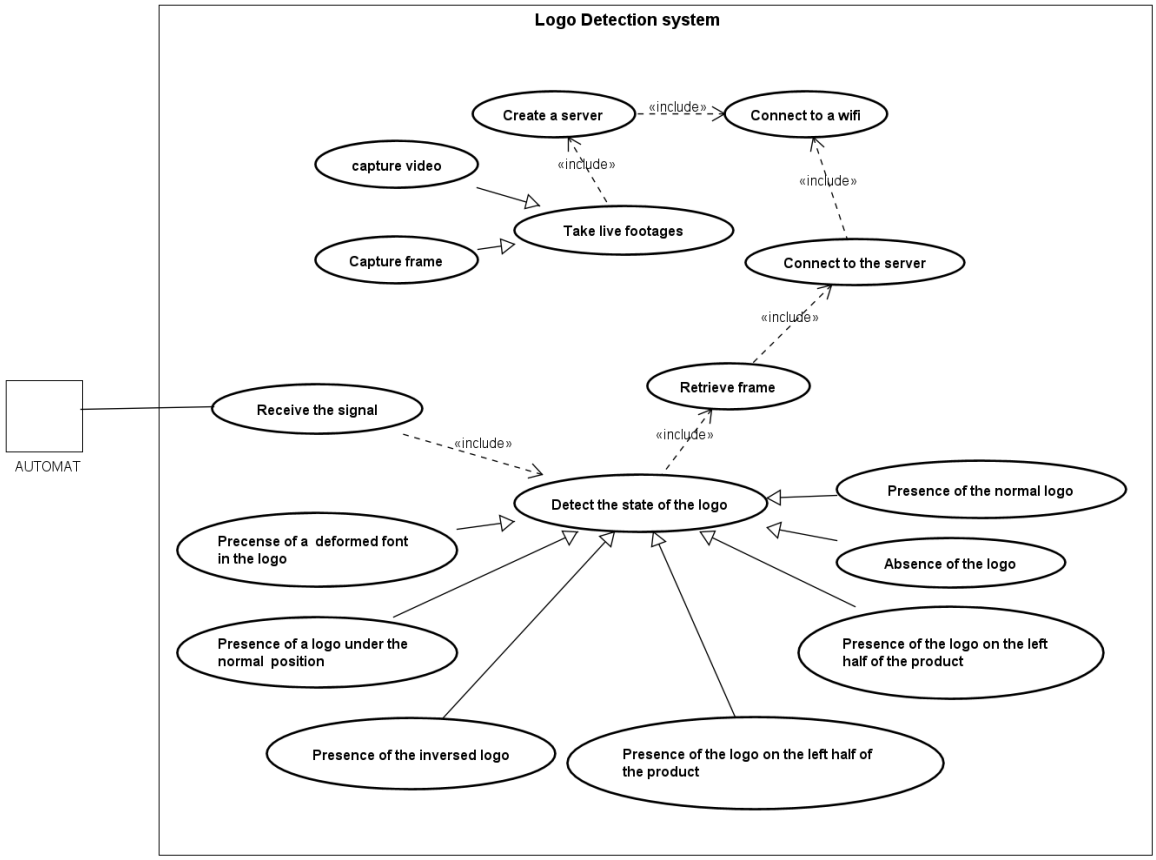



Figure 4.1: Use case diagram version 2

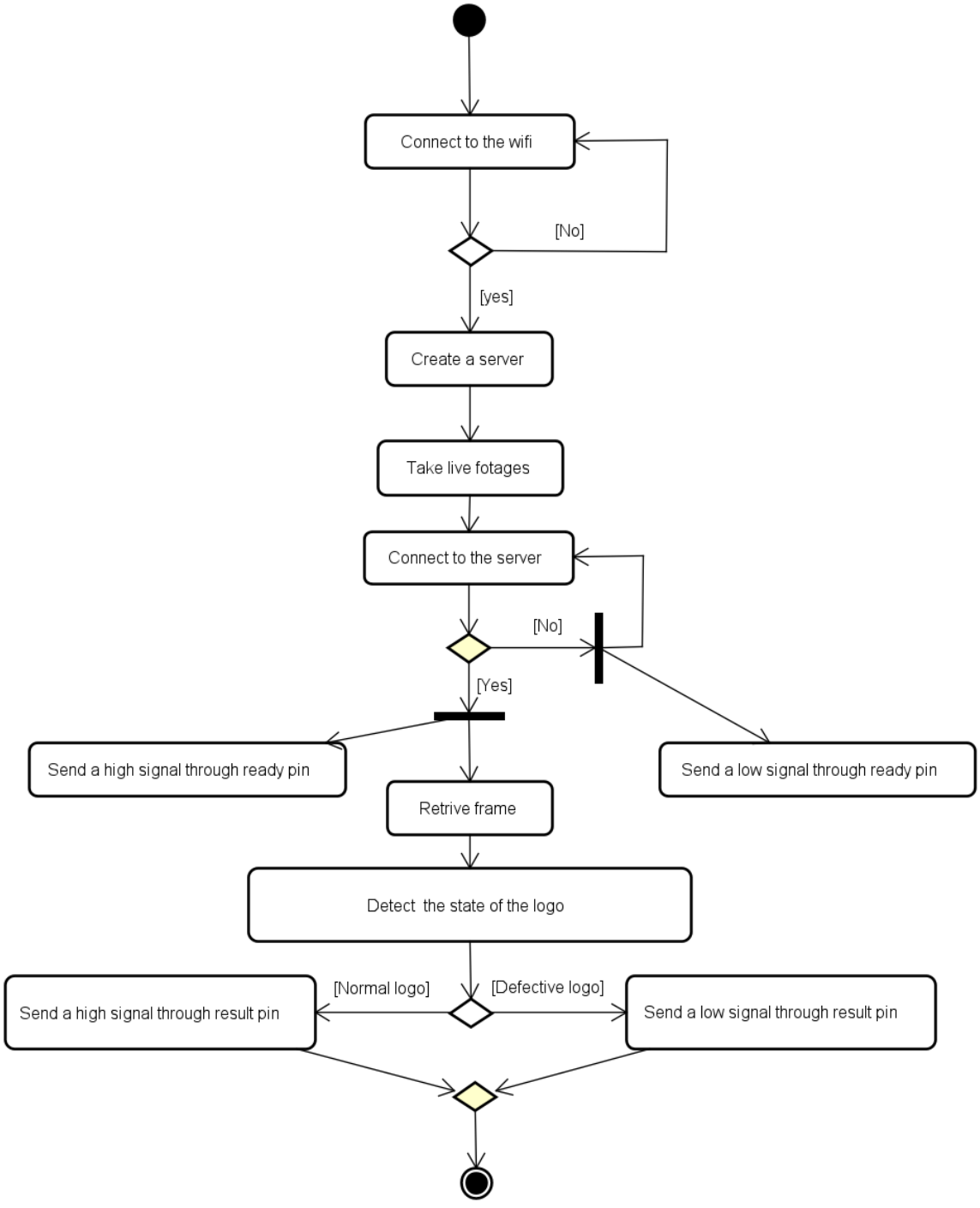



Figure 4.2: Activity diagram version 2

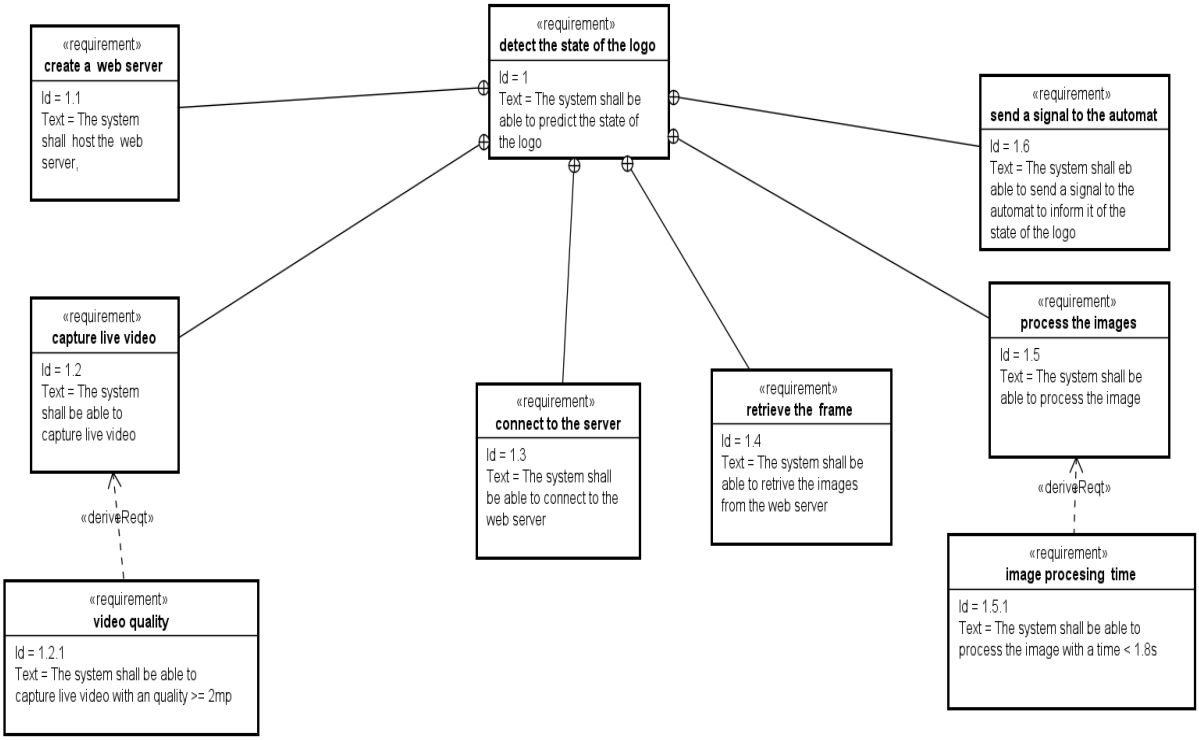



Figure 4.3: Requirement diagram version 2

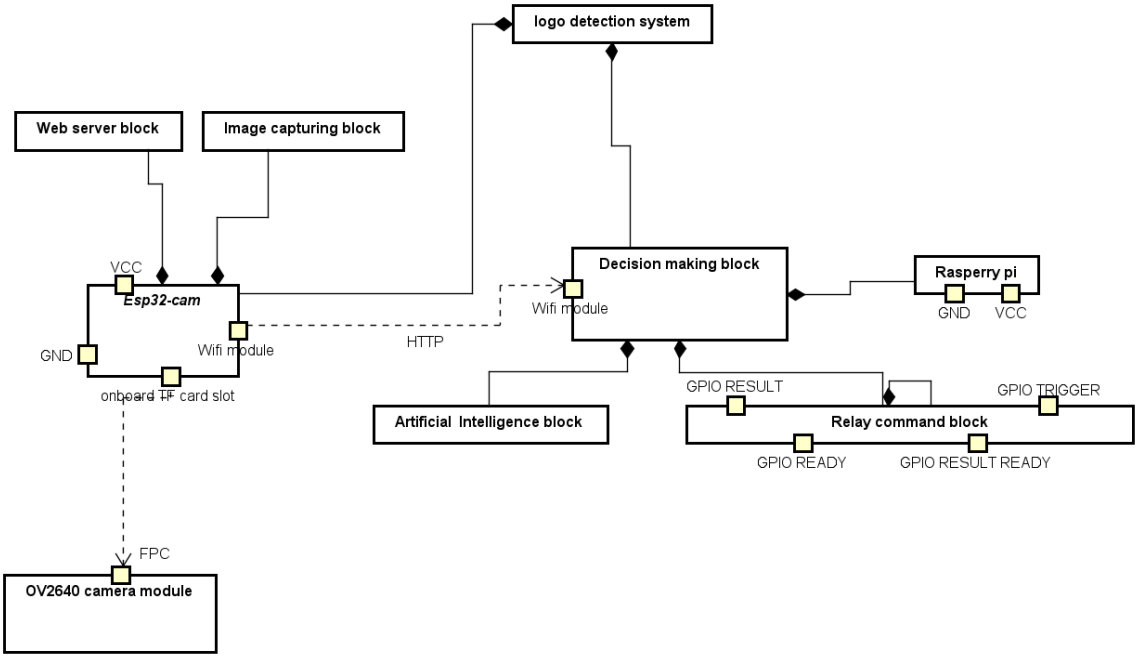



Figure 4.4: Block Definition Diagram version 2

# IV Workflow

## 1 Network improvements

### 1.1 Turning the raspberry pi into an access

There are several reasons I turned the Raspberry Pi into an access point rather than relying on a normal router. First, Flexibility: By setting up a Raspberry Pi as an access point, you have more control over your network and can customize it to meet your specific needs. Second, Cost: Using a Raspberry Pi as an access point can be cheaper than buying a dedicated router, especially if you already have a spare Raspberry Pi lying around.Third,Portability: A Raspberry Pi is small and portable, making it easy to set up an access point on the go. This can be useful if you need to quickly set up a network in a remote location, such as a camping trip or a field research project.Finally, Integration: If I'am already using a Raspberry Pi for other purposes, so I can easily integrate the access point functionality into your existing setup.[9]

### 1.2 Giving static address to ESP32-CAM

There are several advantages of assigning a static IP address to an ESP32-CAM device compared to using a dynamic IP address obtained from a DHCP server. Stability is one of the key advantages. With a static IP address, the ESP32-CAM device will always have the same IP address, ensuring stability in the network connection. In contrast, a dynamic IP address may change periodically, causing connectivity issues if the device's IP address changes without the network administrator's knowledge.[23]

Another advantage is easier network access. By assigning a static IP address, it becomes easier to access the ESP32-CAM device remotely. Since the IP address remains constant, there is no need to check the DHCP server for the device's current IP address. This can be especially beneficial in

situations where frequent access to the device is required.[23]

It is essential for the ESP32-CAM device to have a static IP address. This is because the Raspberry Pi needs to connect to the ESP32-CAM's web server to retrieve images. By assigning a static IP address to the ESP32-CAM, such as 10.42.0.58, the Raspberry Pi can reliably establish a connection with the ESP32-CAM's web server without having to constantly check for the device's IP address.[23]

By including this in my esp32-cam code , I can ensure that the it will always uses the specified static IP address when connecting to the network. This allows the Raspberry Pi to easily retrieve images from the ESP32-CAM's web server by accessing the fixed IP address (10.42.0.58 in this case). This configuration provides stability and reliability to the image retrieval process, enabling seamless communication between the Raspberry Pi and the ESP32-CAM.

# 2 Software improvement

## 2.1 Automating the main script for inference

Once we installed our product in the factory, we noticed that the main script inside it required human intervention to work properly. This was problematic because it meant that an operator had to be present to initiate the script every time the system was started up. We knew that this was not an efficient or sustainable solution, so we began exploring ways to automate the script.

After some research and testing, we were able to configure the script to run automatically at the start of the Raspberry Pi. This allowed the system to operate without any human intervention, reducing the need for manual input and increasing the overall efficiency of the system. With the script now automated, the system could start up and run smoothly without any manual intervention, freeing up our operators to focus on other tasks.

The benefits of automating a script are numerous. Automating a script can reduce the need for manual input, saving time and increasing efficiency. In addition, automating a script can reduce the potential for human error, ensuring that the system operates reliably and accurately. Furthermore, automating a script can help to reduce costs associated with hiring additional staff or training existing staff on manual processes.

Overall, the automation of our script was a significant improvement to our system, and we are confident that it will continue to improve the efficiency and reliability of our product moving forward. By automating this process, we have created a more sustainable and efficient system that will benefit both our team and our end-users.

## 2.2 Detection improvements


Figure 4.5: Logo down from its normal position


Figure 4.6: messed-up fonts in the logo

A new issue has emerged in logo detection where logos can appear in abnormal positions in figure 4.5 or with messed-up fonts in figure 4.6.To address these challenges, a proactive solution was implemented by adding an additional class called "part" to the dataset and retraining the model.

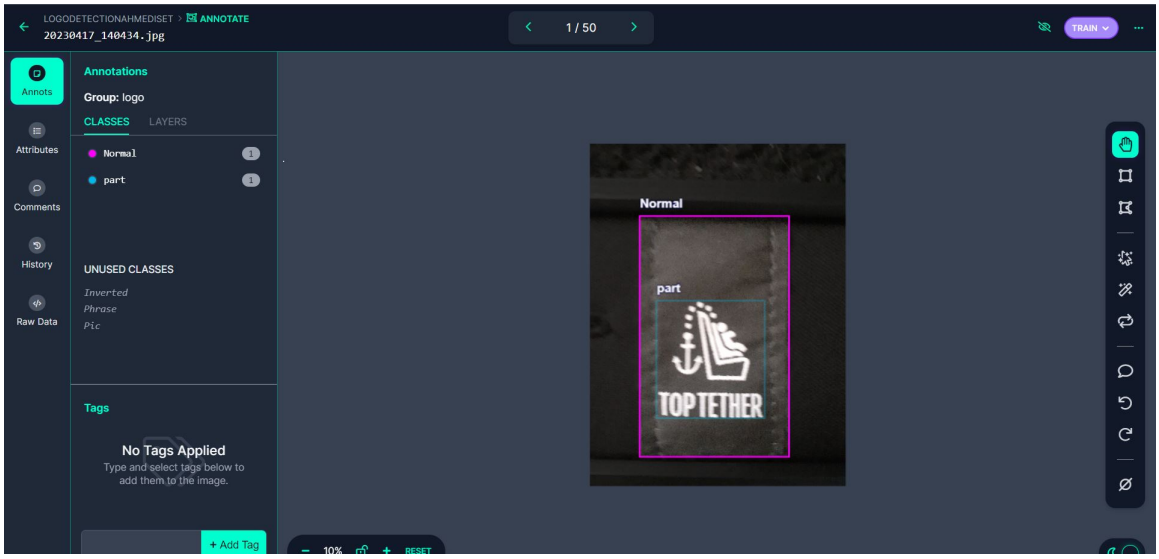



Figure 4.7: part class's annotation

I chose to incorporate the "part" class into my dataset because I faced a unique challenge where I did not have physical access to the actual product. Instead, the only resources provided to me were photographs of the product. This constraint prevented me from capturing images of the physical product from different angles or under various conditions.

In this scenario, I decided to add the label "part" in the available photographs of the normal class since it represented as portions or components of the product in the figure 4.7 . Although they did not encompass the complete product, they were the closest visual representation I had access to. By labeling these images as the "part" class, I intended to train the model to recognize and detect that specific component within the product logo.

While this approach may not have provided direct access to the physical product, it allowed me to leverage the available photographs to train the model effectively. By incorporating the "part" class, I could capture the visual characteristics of the actual logo present in the provided images.


Figure 4.8: Object detection of a mess-up font in the logo

First, I calculated the height of the bounding boxes for the ”normal” and ”part” classes in figure 4.8. Then, I determined the ratio by dividing the height of the ”part” bounding box by the height of the ”normal” bounding box. After conducting extensive experiments, I established a threshold of 0.51 as the ideal ratio. If the calculated ratio (hPart / hNormal) was greater than 0.51, it indicated that the logo's font was distorted. I noticed that the ”normal” logo generally had a smaller height compared to the logo with the distorted font.


Figure 4.9: Object detection on a logo with a position problem

For the second problem, I utilized the y coordinates obtained from each bounding box in figure 4.9. I developed this formula:

$$ratio\_position = (y1 - ny1) * 100//(ny2 - ny1) \tag{4.1}$$

where y1 represents the y-coordinate minimum of the "part" bounding box, ny1 represents the minimum y-coordinate of the "normal" bounding box, and ny2 represents the maximum y-coordinate of the "normal" bounding box. Subsequently, I set a threshold of 40 based on numerous experiments. If the calculated ratio_position was greater than 40, it indicated that the logo was positioned below the normal location.

These approaches helped me address the challenges related to the logo's position and font during the object detection task.

## 3 Fixing the temperature issue

After installing the first version of our product in the factory, we noticed that the temperature of the Raspberry Pi was rising significantly. This was likely due to the hot environment and the fact that the Raspberry Pi was running YOLOv5, which requires a significant amount of processing power. We knew that this issue could impact the reliability and longevity of our product, so we immediately began exploring solutions.

After considering several options, we decided to install a fan to cool the Raspberry Pi. This solution proved to be effective in reducing the temperature of the Raspberry Pi and maintaining stable performance. By keeping the Raspberry Pi cool, we were able to prevent it from overheating and potentially causing damage to the system. Additionally, we found that cooling the Raspberry Pi improved the overall performance of our system, allowing it to run more smoothly and efficiently.

In general, the benefits of cooling a Raspberry Pi are numerous. Cooling a Raspberry Pi can improve its longevity, reduce the risk of damage, and ensure that it runs smoothly and efficiently. Overheating can cause a Raspberry Pi to crash, freeze, or even become permanently damaged. By keeping the Raspberry Pi cool, we can prevent these issues and ensure that our product is reliable and

durable. Furthermore, cooling a Raspberry Pi can help to extend its lifespan, which can ultimately save time and money in the long run.

# V Conclusion

In conclusion, the meticulous and comprehensive modifications implemented in this chapter have astoundingly transformed and amplified the functionality and usability of our project, propelling it to unparalleled levels of excellence.

# Chapter 5

# PCB Board and Case Design

## I Introduction

In this chapter, we focus on the crucial aspects of designing a PCB (Printed Circuit Board) card and a case to hold our system. These components play a pivotal role in ensuring the robustness, efficiency, and overall user experience of our project. Through careful planning and innovative approaches, we aim to create a seamless integration between hardware and aesthetics, providing a reliable and visually appealing solution for our users' needs.

## II PCB board design

There are several compelling reasons why using a printed circuit board (PCB) to mount and connect components is preferred over relying solely on a breadboard and wires. Firstly, a PCB offers enhanced reliability and stability compared to a breadboard. The components are securely soldered onto the PCB, ensuring a more secure and permanent connection. This eliminates the risk of loose connections or accidental disconnections that can occur with breadboards and wires, especially in situations where the device may be subjected to movement or vibrations.[29]

Secondly, a PCB provides a more compact and organized layout for the components. With a breadboard, components and wires are often scattered and loosely arranged, making it challenging to maintain a clean and efficient design. PCBs allow for a streamlined and optimized placement of components, reducing the overall size of the circuit and facilitating easier troubleshooting and maintenance.[29]

Furthermore, PCBs offer improved electrical performance and reduced signal interference. The copper traces on a PCB provide precise pathways for electrical signals, minimizing signal loss and crosstalk. This is particularly crucial in high-frequency applications or circuits that require precise signal transmission. In contrast, breadboards and wires introduce additional resistance and capacitance, which can degrade signal quality and introduce noise.[29]

Another significant advantage of using a PCB is scalability and mass production. Once a circuit design is finalized and tested on a breadboard, transitioning to a PCB layout allows for easy replication and mass production. PCB manufacturing processes enable consistent and efficient production of multiple identical circuits, making it ideal for commercial or large-scale applications.[29]

Lastly, PCBs offer a level of professionalism and aesthetics to the final product. The neatly arranged components and the absence of visible wires contribute to a clean and polished appearance. This is particularly relevant when designing devices for presentations, demonstrations, or commercial purposes, where visual appeal can significantly impact the perception of the product's quality and reliability.[29]

In conclusion, using a PCB to mount and connect components offers numerous advantages over relying solely on a breadboard and wires. The enhanced reliability, compact layout, improved electrical performance, scalability, and professional aesthetics make PCBs the preferred choice for creating robust and efficient electronic circuits.

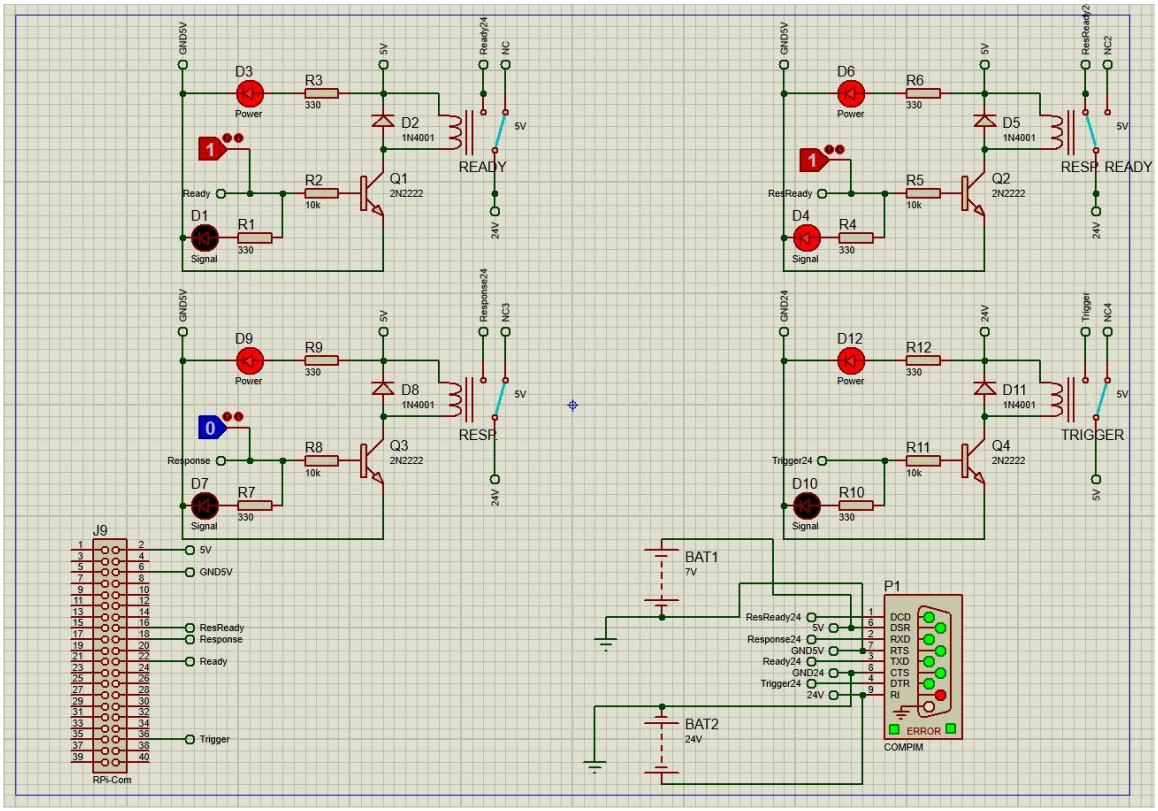



Figure 5.1: Schematic Capture of the Interface card

Using Proteus, I began by designing the circuit in the schematic stage. The software provided an intuitive interface that allowed me to depict the components, their interconnections, and the overall circuit architecture. This schematic representation in figure 5.1 served as a blueprint for the circuit's functionality and ensured accurate translation into the physical layout.

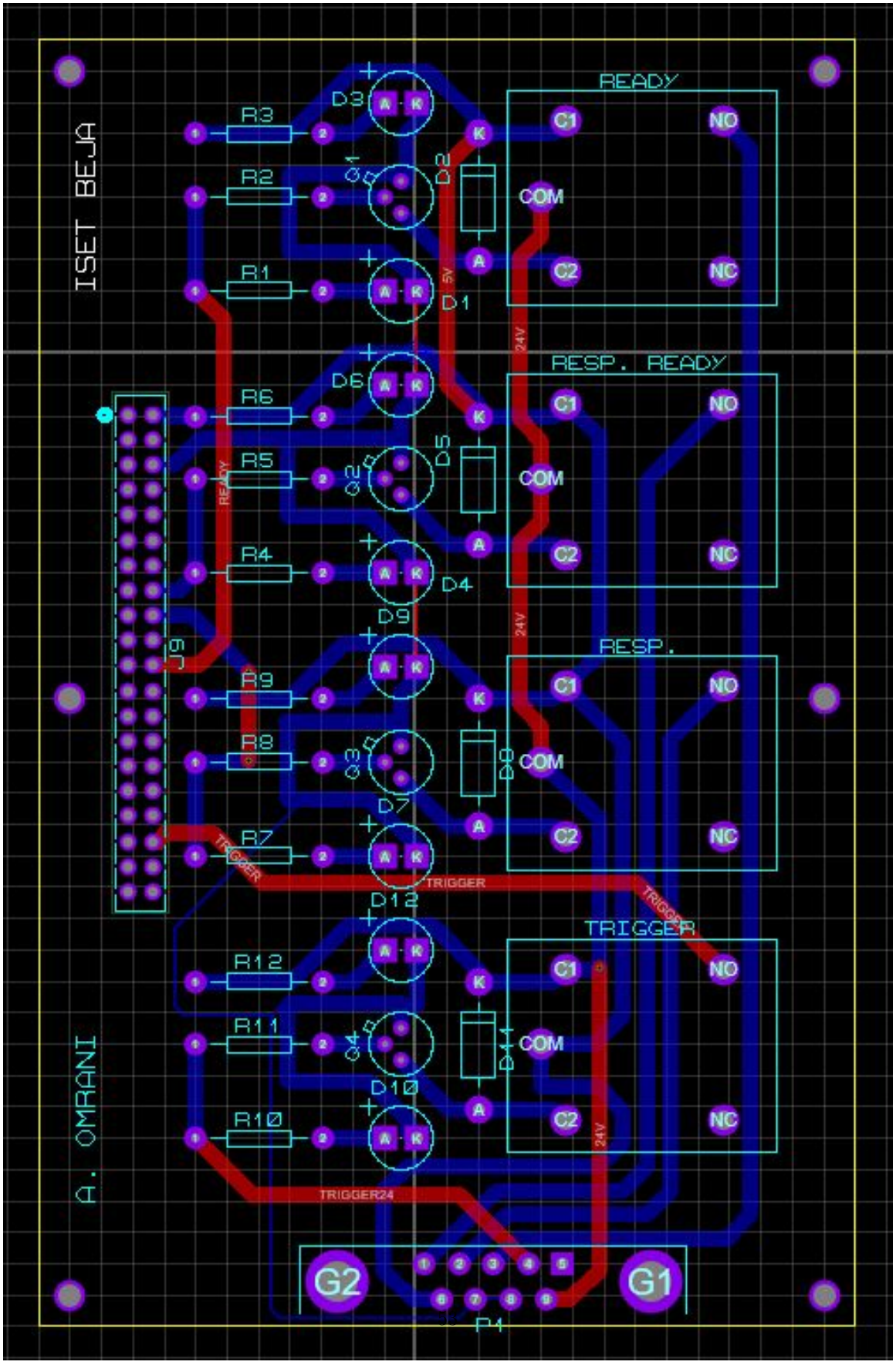



Figure 5.2: PCB layout of the Interface card

With the schematic design in place, I seamlessly transitioned to the PCB layout stage using Proteus as you can see in figure 5.2. The software offered a user-friendly environment where I could place components, define their physical arrangement on the PCB, and route traces to establish the required electrical connections. Proteus also allowed me to define design rules and constraints, ensuring proper spacing, clearance, and signal integrity. In conclusion, by leveraging Proteus, I successfully designed and transformed the circuit from a schematic to a PCB layout. The software's capabilities, including schematic design, component placement, and trace routing, enabled me to create a well-structured and functional PCB design. Proteus enhanced the overall efficiency, accuracy, and reliability of the design process, resulting in a PCB ready for fabrication.

# III Conclusion

In conclusion, the meticulous design and implementation of the PCB card and case in this chapter have significantly contributed to the success of our project. The PCB card serves as the backbone of our system, enabling efficient and reliable electrical connections between various components. Simultaneously, the thoughtfully designed case acts as a protective shield, safeguarding the internal elements from external factors while offering an aesthetically pleasing appearance.

# Chapter 6

# Annex

## I Project component

### 1 Hardware environment

#### 1.1 Raspberry Pi 3 Model B+

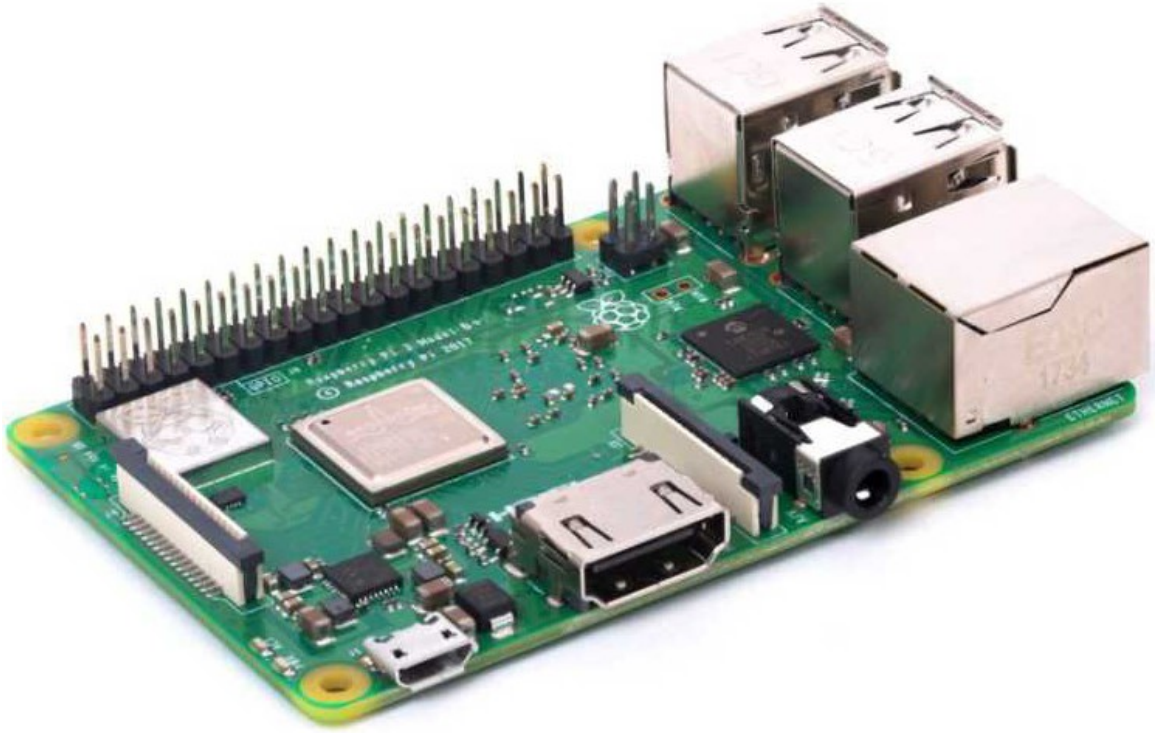


Figure 6.1: Raspberry Pi 3 Model B+

| Raspberry Pi 3 Model B characteristics [25] | |
|---|---|
| **Component** | **Specification** |
| Processor | 1.4 GHz quad-core BCM2837B0 ARMv8 64bit CPU |
| Memory | 1 GB RAM |
| Connectivity | -dual-band (2.4 GHz and 5 GHz) IEEE 802.11.b/g/n/ac wireless LAN (WiFi)<br>- LAN, Bluetooth 5.0, BLE<br>- Gigabit Ethernet<br>- 4 × USB 2.0 ports |
| GPIO | Standard 40-pin GPIO header (fully backwards-compatible with previous boards) |
| Video & Sound | - full size HDMI<br>- MIPI DSI display port<br>- MIPI CSI camera port<br>- 4-pole stereo audio and composite video port |
| Multimedia | -MPEG-4 decode (1080p30)<br>- H.264 encode(1080p30)<br>- OpenGL ES 1.1, 2.0 graphics |
| SD Card Support | Micro SD format for loading operating system anddata storage |
| Input Power | - 5V/2.5A DC via micro USB connector<br>- 5V DC via GPIO header (minimum 3A)<br>-Power over Ethernet (PoE)–enabled (requires separate PoE HAT) |
| Environment | Operating temperature 0–50$^{o}$C |

## 1.2 Raspberry Pi 4 Model B

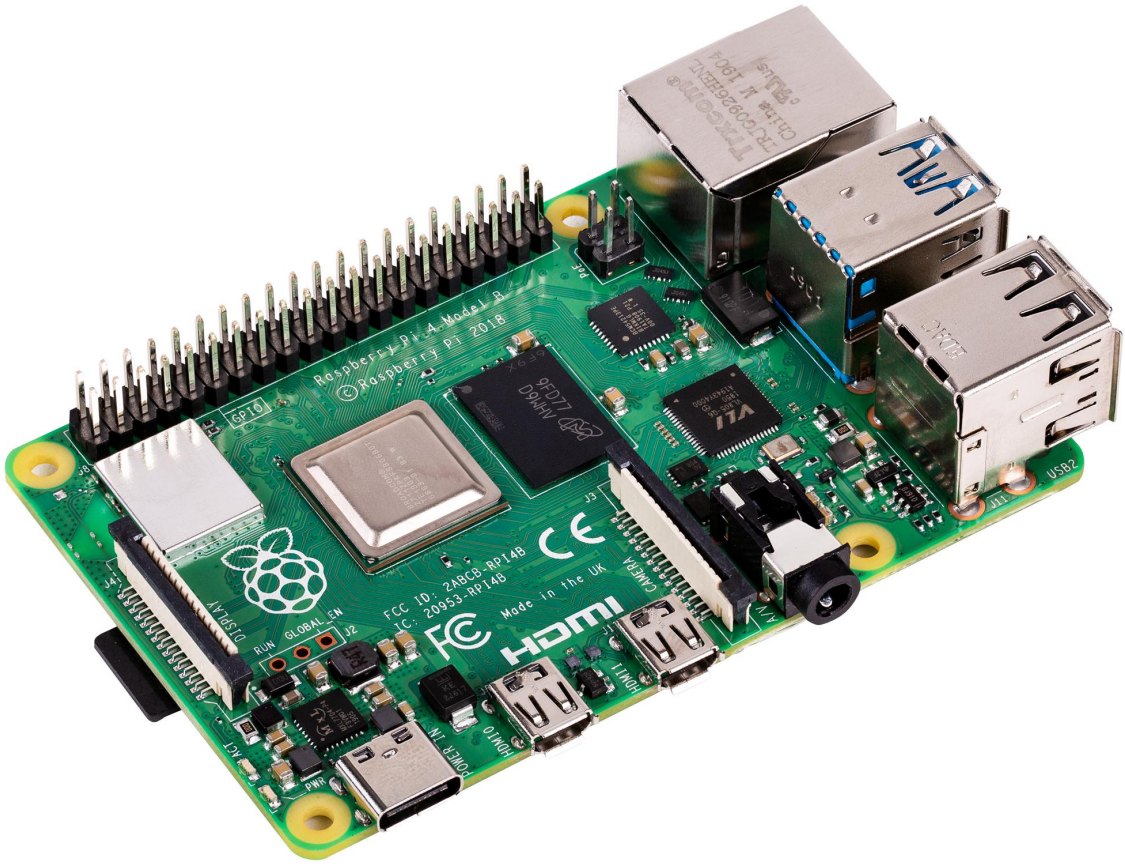



Figure 6.2: Raspberry Pi 4 Model B

| Raspberry Pi 4 model B characteristics [24] | |
|---|---|
| **Component** | **Specification** |
| Processor | Broadcom BCM2711, quad-core Cortex-A72 (ARM v8) 64-bit SoC @ 1.5GHz |
| Memory | 1GB, 2GB or 4GB LPDDR4 (depending on model) |
| Connectivity | - 2.4 GHz and 5.0 GHz IEEE 802.11b/g/n/ac wireless<br>- LAN, Bluetooth 5.0, BLE<br>- Gigabit Ethernet<br>- 2 × USB 3.0 ports<br>- 2 × USB 2.0 ports |
| GPIO | Standard 40-pin GPIO header (fully backwards-compatible with previous boards) |
| Video & Sound | - 2 × micro HDMI ports (up to 4Kp60 supported)<br>- 2-lane MIPI DSI display port<br>- 2-lane MIPI CSI camera port<br>- 4-pole stereo audio and composite video port |
| Multimedia | - H.265 (4Kp60 decode)<br>- H.264 (1080p60 decode, 1080p30 encode)<br>- OpenGL ES, 3.0 graphics |
| SD Card Support | Micro SD card slot for loading operating system and data storage |
| Input Power | - 5V DC via USB-C connector (minimum 3A)<br>- 5V DC via GPIO header (minimum 3A)<br>- Power over Ethernet (PoE)–enabled<br>(requires separate PoE HAT) |
| Environment | Operating temperature 0–50ºC |

### 1.3 ESP32-CAM

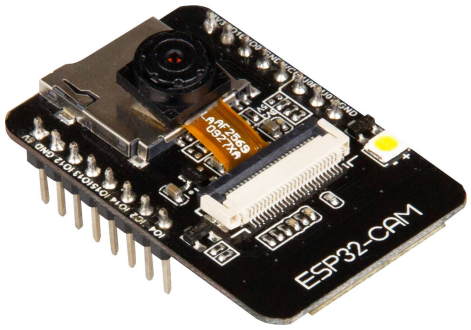



Figure 6.3: ESP32-CAM

| ESP32-CAM characteristics [12] | |
|---|---|
| **Component** | **Specification** |
| WiFi+Bluetooth module | ESP-32S |
| Camera module | OV2640 2MP |
| SPI Flash | 4MB |
| RAM | Internal 512KB + External 4MB PSRAM |
| Onboard TF card slot | Supports up to 4G TF card for data storage |
| Wi-Fi | 802.11b/g/n/e/i |
| Operating voltage | 3.3/5 Vdc |
| Power consumption (Flash off) | 180mA@5V |
| Power consumption (Flash on and brightness max) | 310mA@5V |
| Power consumption (Modern-Sleep) | as low as 20mA@5V |
| Power consumption (Light-Sleep) | as low as 6.7mA@5V |
| Power consumption (Deep-Sleep) | as low as 6mA@5V |
| Operating temperature | -20 °C  85 °C |
| Dimensions | 40.5mm x 27mm x 4.5mm |
| Flash light | LED built-in on board |

### 1.4 PC

Figure 6.4: lenovo ideapad gaming 3

| Specifications of the PC | |
|---|---|
| **Component** | **Specification** |
| Processor | AMD Ryzen 7 4800H |
| Memory | 16 GB DDR4 RAM |
| Graphics | NVIDIA GeForce GTX 1650Ti (4GB GDDR6) |
| Storage | 512 GB SSD |
| Operating System | Windows 10 |

# 2 Software environment

## 2.1 Software

- Arduino IDE

Figure 6.5: Arduino IDE Logo

**Arduino IDE** is an official Arduino software application used for writing, compiling, and uploading code to Arduino microcontrollers. The IDE environment consists of two basic parts: Editor and Compiler and supports both C and C++ languages.[5]

- Jupter Notebook

Figure 6.6: Jupyter notebook logo

**Jupyter Notebook** is a free, open-source web application that enables interactive computing and data analysis using various programming languages, including Julia, Python, and R.[6]

It allows users to create virtual lab notebooks to support workflows, code, data, and visualizations detailing the research process, making science more open and accessible.[6]

- Raspberry Pi Imager

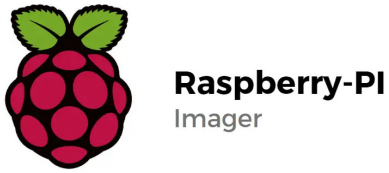



Figure 6.7: Raspberry pi imager logo

**Raspberry Pi Imager** is a free and open-source software application developed by the Raspberry Pi Foundation.[14] With Raspberry Pi Imager, users can easily choose to install a variety of os, like Raspbian, Ubuntu, Kali Linux, and more, onto a microSD card that can be used to boot up your Raspberry Pi.

It can be downloaded and installed on Windows, Mac, and Linux operating systems.[14]

- PuTTY

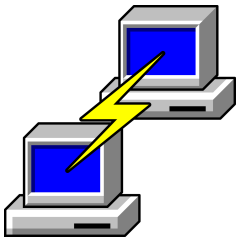


Figure 6.8: puttyLogo

**PuTTY** is a free and open-source terminal emulator, serial console, and network file transfer application. It was originally developed for Windows but is now available on many other operating systems.[15]

PuTTY also supports many network protocols, such as Telnet, rlogin, SSH and raw TCP. It also includes additional features such as session management, SSH key generation, and support for local printing.[15]

- VNC

Figure 6.9: VNCLogo

**VNC** (Virtual Network Computing) is a thin-client system that allows users to remotely control and operate another computer or server over a network. It consists of two components: a server that runs on the remote computer, and a client that runs on the local computer.[16]

The server sends screen updates to the client, so the user can see and interact with the remote computer's desktop as if they were sitting right in front of it.[16]

- Geany

Figure 6.10: Geany logo

**Geany** is a simple and lightweight text editor designed for programmers and developers. It provides features such as syntax highlighting for a wide range of programming languages, code folding, auto-indentation, and built-in support for various programming tools and compilers.[17]

- Google Colab

Figure 6.11: Google colab logo

**Google Colab** (short for Collaboratory) is a cloud-based platform provided by Google that allows users to run and share Jupyter notebook files for data analysis, machine learning and deep learning tasks. .[18]

It provides access to computing resources such as CPU, GPU,RAM, disk and TPU for free, allowing users to execute complex computational tasks without the need to use their local hardware..[18]

### 2.2 programming languages

- C++

Figure 6.12: C++ logo

**C++** is a programming language that is widely used in software development. It is a standardized, general-purpose, and object-oriented language, which means it can be used to create a variety of applications, including system software, device drivers, video games, and desktop applications.[7]

- Python

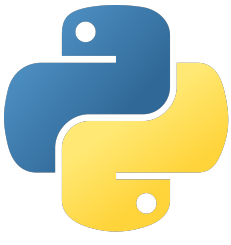


Figure 6.13: Python logo

**Python** is a popular high-level programming languagethat supports object-oriented, functional, and imperative programming styles. scripting language, but can be compiled into computer-readable binary.[7]

# Bibliography


[1] Eugene Dorfman, 2022. *How AI for Quality Control Enhances Yield in Manufacturing.*

[2] Benjamin Wann, 2022. *Why Do Manufacturers Struggle with Profitability?.*

[3] David Greenfield, 2020. *Why Do Manufacturers Struggle with Profitability?.*

[4] Lenny Delligatti , 2013. *SysML Distilled A Brief Guide.*

[5] Mohamed FEZARI and Ali Al Dahoud, 2018. *Integrated Development Environment "IDE" For Arduino.*

[6] Bernadette M. Randles, Milena S. Golshan, Irene V. Pasquetto and Christine L. Borgman, 2017. *Using the Jupyter Notebook as a Tool for Open Science: An Empirical Study.*

[7] Slobodan Dmitrović, 2020. *Modern C++ for Absolute Beginners: A Friendly Introduction to C++ Programming Language and C++11 to C++20 Standards.*

[8] Prasanna Sattigeri, 2019. *Deep Learning Inference: A Comprehensive Overview.*

[9] Alex Gizis, 2019. *5 Uses for Raspberry Pi Routers to Get Better Internet.*

[10] Chakraborty D.,2021. *OpenCV Contour Approximation ( cv2.approxPolyDP ),.*

[11] Willow Garage, 2010. *OpenCV Reference Manual v2.2.*

[12] Handson Technology. *ESP32-CAM WiFi+Bluetooth+Camera Module Datasheet ),.*

[13] Espressif,2019. *ESP32-WROOM-32 Datasheet),.*

[14] Raspberry Pi Software,(n.d.). *Retrieved from the raspberrypi website*

[15] PuTTY. (n.d.). *Retrieved from the PuTTY website*

[16] RealVNC. (n.d.). *what is vnc remote access technology*

[17] Geany. (n.d.). *Retrieved from the geany website*

[18] Google Research . (n.d.). *Retrieved from Google Research website*

[19] Pappu Kumar YadavJ. Alex ThomassonStephen W. SearcyRobert G. HardinUlisses Braga-NetoSorin C. PopescuDaniel E. MartinRoberto RodriguezKarem MezaJuan EncisoJorge Solórzano DiazTianyi Wang, 2021. *Assessing The Performance of YOLOv5 Algorithm For Detecting Volunteer Cotton Plants in Corn Fields at Three Different Growth Stages.*

[20] Kathrin Blagec , Georg Dorffner , Milad Moradi , Matthias Samwald , 2020. *A critical analysis of metrics used for measuring progress in artificial intelligence .*

[21] Marko Horvat, Gordan Gledec, 2022. *A comparative study of YOLOv5 models performance for image localization and classification* .

[22] ultralytics, 2023. *Retrieved from yolov5 github repository*

[23] hitechwhizz, 2022. *Drawbacks and Benefits of Static IP Address*

[24] Raspberry Pi Trading Ltd, 2019. *Raspberry Pi 4 Model B datasheet*

[25] Raspberry Pi Trading Ltd, 2016. *Raspberry Pi 3 Model B+ datasheet*

[26] TESCA, 2023. *Retrieved from TESCA webiste*

[27] Ifm efector,nd. *IFM O2D220 datasheet*

[28] Niklas Donges,2022. *What Is Transfer Learning? Exploring the Popular Deep Learning Approach.*

[29] Apollotechnical,nd. *THE IMPORTANCE OF PRINTED CIRCUIT BOARDS FOR ELECTRONICS*