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

Academic year 2019-2020

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

Introduction 1

1 Project Framework and Methodology **2**
I Presentation of the enterprise 2
1 description of the enterprise 2
2 description of the enterprise's services 2
3 administrative organization chart of the enterprise 2
II Study of the existing system 2
1 description of the existing system 2
2 criticism of the existing system 2
3 administrative organization chart of the enterprise 2
III Working methodology and modeling language 2
1 Traditional work methodology vs Agile method 2
2 Chosen methodology: SCRUM 3
2.1 Reasons 3
2.2 The Scrum team 4
2.3 Other terminologies 4
3 Modeling languages : UML and SysMl 4
3.1 UML 4
3.2 SysMl 5
IV Project management with SCRUM 5
1 Team and roles 5
2 Backlog Product 5

2 Sprint 0: Logo Detection with Image Processing **7**
I Specification of requirements 7
1 actors identification 7
2 Description of functional requirements 7
3 Description of non-functional requirements 7
II Modeling languages diagrams 8
1 UML: use case diagram 8
2 SysMl 8
III Project component 8
1 Hardware and Software environment 8
IV Workflow 9
0.1 Brightness and contrast adjustment 9
0.2 reflection 10
0.3 grayscaling 11
0.4 reflection 12
0.5 Image thresholding 12

0.6 reflection . . . 14
0.7 Contour detection . . . 14

3 Sprint Two: Raspberry pi 4 and ESP32 Cam Integration for Logo Detection 17
I Specification of requirements . . . 17
1 actors identification . . . 17
2 Description of functional requirements . . . 17
3 Description of non-functional requirements . . . 17
II Modeling languages diagrams . . . 18
1 SysMl . . . 18
III Project component . . . 20
1 Hardware and Software environment . . . 20
IV Workflow . . . 20
0.1 Setting up the rapsberry pi . . . 20
0.2 Deep learning pipeline . . . 25

4 Sprint 2: Stabilizing the System 34

5 Sprint 4: PCB Board and Case Design 35

6 Annex 36
I Project component . . . 36
1 Hardware environment . . . 36
1.1 ESP32-CAM . . . 36
1.2 PC . . . 37
2 Software environment . . . 37
2.1 Software . . . 37
2.2 programming languages . . . 40

List of Figures

1.1 Scrum process 3

2.1 Use case Diagram for sprint version 0 8
2.2 Phone's images : normal to darker 9
2.3 esp32-cam on printed images : normal to darker 9
2.4 esp32-cam on the actual product : normal to darker 10
2.5 Phone's images :darker to gray 11
2.6 esp32-cam on printed images : darker to gray 11
2.7 esp32-cam on the actual product : darker to gray 12
2.8 Phone's images :Gray to binary 13
2.9 esp32-cam on printed images : Gray to binary 13
2.10 esp32-cam on the actual product :Gray to binary 14
2.11 prediction results 15

3.1 Use case Diagram for sprint version 1 19
3.2 putty interface 22
3.3 Update.sh file 22
3.4 Enable vnc menu 23
3.5 prepareEnv.sh file 24
3.6 Comparison between yolov5 family and Efficientdet for the COCO dataset in term of speed and accuracy 26
3.7 Normal dataset 27
3.8 Inverted dataset 27
3.9 Labeling an image 28
3.10 Augmented dataset 29

6.1 ESP32-CAM 36
6.2 lenovo ideapad gaming 3 37
6.3 Arduino IDE Logo 38
6.4 Jupyter notebook logo 38
6.5 Raspberry pi imager logo 39
6.6 puttyLogo 39
6.7 VNCLogo 39
6.8 Geany logo 40
6.9 Google colab logo 40
6.10 C++ logo 41
6.11 Python logo 41

List of Tables

1.1 Comparison of heavyweight and agile methods [2] 3
1.2 SCRUM Actors 6

Execution time of sh update.sh 23
Execution time of prepareEnv.sh 25
Comparison of YOLOv5 Models [21] 26
Training and Validation Metrics for YOLOv5s 31
Training and Validation Metrics for YOLOv5s 32
Training and Validation Metrics for YOLOv5s 32

6.1 ESP32-CAM characteristics [10] 37
6.2 Specifications of the PC 38

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.

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.

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.

At TOP TETHER, 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

Introduction

I Presentation of the enterprise

1 description of the enterprise
2 description of the enterprise's services
3 administrative organization chart of the enterprise

II Study of the existing system

1 description of the existing system
2 criticism of the existing system
3 administrative organization chart of the enterprise

III Working methodology and modeling language

1 Traditional work methodology vs Agile method

In order to ensure a successful outcome for an IT project, it is crucial to follow a structured work methodology throughout its entire lifecycle. A work methodology provides a framework for project management, which includes defining tasks, assigning responsibilities, monitoring progress, and ensuring that deadlines are met.[2]

There are many different methodologies that can be used for IT project management, such as Agile[1] and Heavyweight[1] . Each methodology has its own set of principles, practices, and tools that are designed to optimize project performance and efficiency.

	Heavyweight Methods	Agile Methods
Approach	Adaptive	Predictive
Success Measurement	Business Value	Conformation to plan
Project size	Small	Large
Management Style	Decentralized	Autocratic
Perspective to Change	Change Adaptability	Change Sustainability
Culture	Leadership-Collaboration	Command-Control
Documentation	Low	Heavy
Emphasis	People-Oriented	Process-Oriented
Cycles	Numerous	Limited
Domain	Unpredictable/Exploratory	Predictable
Upfront Planning	Minimal	Comprehensive
Return on Investment	Early in Project	End of Project
Team Size	Small/Creative	Large

Table 1.1: Comparison of heavyweight and agile methods [2]

2 Chosen methodology: SCRUM

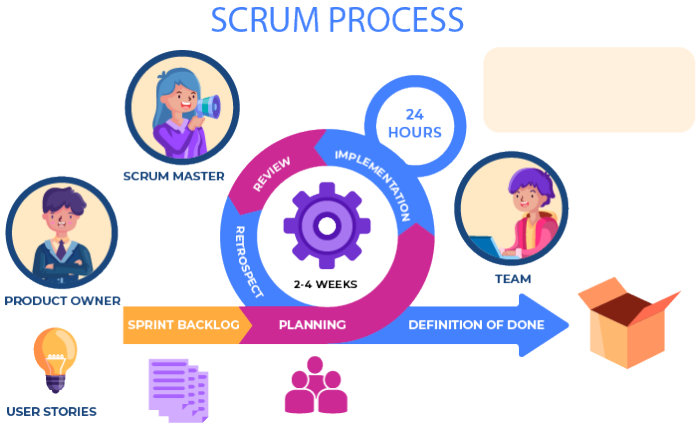


Figure 1.1: Scrum process

2.1 Reasons

The adopted development methodology is Scrum. The choice is based on Scrum's strengths, which can be summarized as follows:

- **Flexibility** : Scrum is designed to be flexible and adaptable to changing project requirements. This allows teams to respond quickly to changes in the project environment.[1][2]
- **Collaboration**: Scrum emphasizes collaboration and communication between team members. [1][2]
- **Continuous Improvement** : Scrum includes regular reviews and retrospectives, which provide opportunities for the team to reflect on their progress and identify areas for improvement.[1][2]
- **Transparency** : Scrum provides a framework for making project progress and status visible to all stakeholders.[1][2]

- **Empowerment**: Scrum encourages team members to take ownership of their work and make decisions independently.[1][2]

2.2 The Scrum team

And we can not end this section without talking about the scrum team which is composed of:

- The **Scrum Master** is responsible for implementing and maximizing the benefits of the Scrum process. They facilitate Scrum events, remove obstacles, and promote continuous improvement.[2]
- The **team** is a cross-functional group of self-motivated individuals who share tasks and responsibilities, working collaboratively to manage themselves and deliver successful product increments every Sprint.[2]
- The **Product Owner** is typically the project's key stakeholder, who has a deep understanding of users, the marketplace, competitors, and trends. They are responsible for managing the Product Backlog to maximize the project's value and representing the interests of everyone with a stake in the project and its resulting product.[2]

2.3 Other terminologies

Let's now look at some options specific to the agile mode and more particularly to SCRUM.

- A **sprint** is a time-boxed period when a SCRUM team works to complete a set amount of work in one repeating cycle, typically lasting no longer than one month and fixed throughout the overall work.[2]
- The **Product Backlog** in SCRUM is a prioritized list of items (or user stories) that represent work needed to deliver a product or service, including estimated completion times. It changes with business conditions or technology.[2]
- Th**User stories** are often used to describe the Product Backlog items in terms of their value to the end user of the product.[2]

3 Modeling languages : UML and SysMl

3.1 UML

UML , or Unified Modeling Language, is a standardized visual language for describing software systems composed of a set of diagrams, each of which provides a different view of the project to be addressed.[3]

UML provides us with diagrams to represent the system to be developed: its operation, startup, actions that can be performed by the system, etc. Creating these diagrams is therefore equivalent to modeling the needs of the software to be developed.[3]

- Structure diagrams
  - Class Diagram
  - Component Diagram
  - Deployment Diagram
  - Object Diagram
  - Package Diagram
  - Profile Diagram
  - Composite Structure Diagram

- Behavioral Diagrams
  - Use Case Diagram
  - Activity Diagram
  - State Machine Diagram
  - Sequence Diagram
  - Communication Diagram
  - Interaction Overview Diagram
  - Timing Diagram

3.2 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]

- 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 Project management with SCRUM

1 Team and roles
2 Backlog Product

Conclusion

Role	Actor	Mission
Team	Ahmed Omrani	Conception, development, unit testing, deployment.
SCRUM master	Radhwen Aloui	Ensure the smooth running of the SCRUM methodology..
Product owner	**********	Define the product features and ensure their conformity.

Table 1.2: SCRUM Actors

Chapter 2

Sprint 0: Logo Detection with Image Processing

Introduction

I Specification of requirements

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

1 actors identification

2 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 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.

3 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.
- **Security:** The system should have appropriate security measures in place to prevent unauthorized access to the servers and data.
- **Scalability:** The system should be able to handle multiple simultaneous connections and requests from clients without compromising its performance.

- **Maintainability:** The system should be easy to maintain and update, with clear and well-documented code and configuration.
- **Compatibility:** The system should be compatible with a wide range of devices and platforms.
- **Usability:** The system should be easy to use and understand for both technical and non-technical users.

II Modeling languages diagrams

1 UML: use case diagram

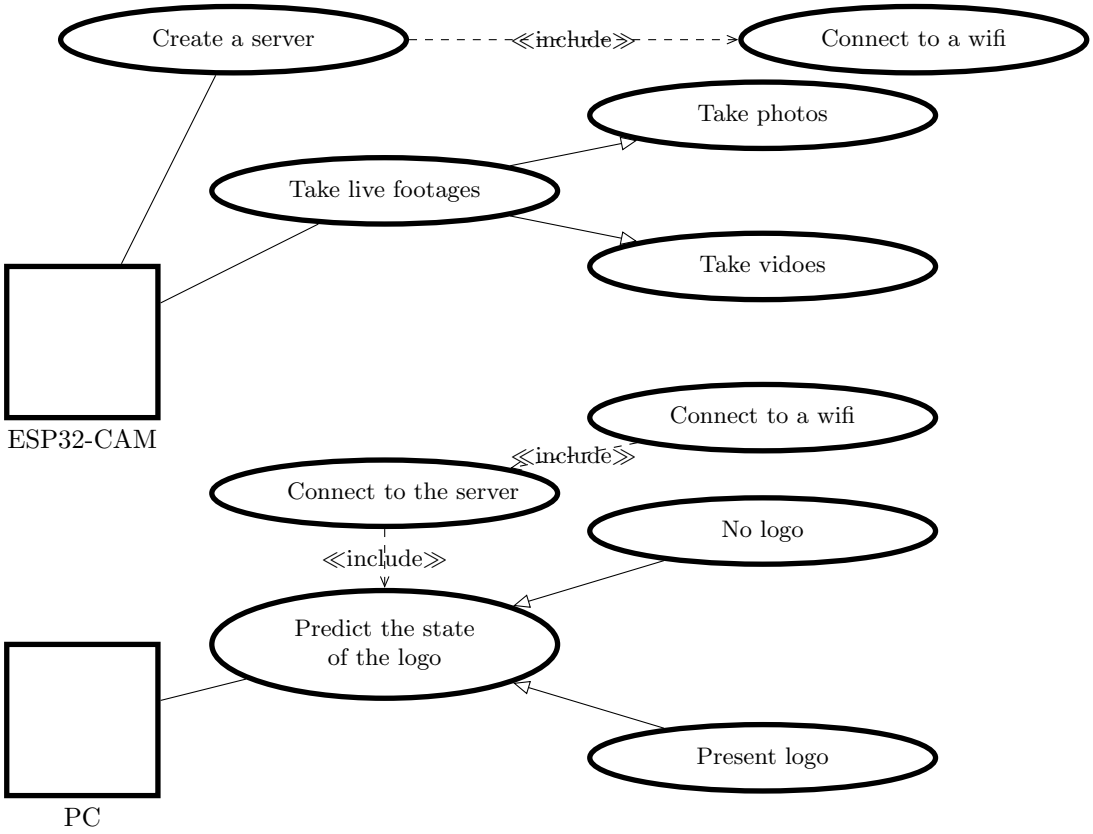


Figure 2.1: Use case Diagram for sprint version 0

2 SysMl

III 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.

IV 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

0.1 Brightness and contrast adjustment

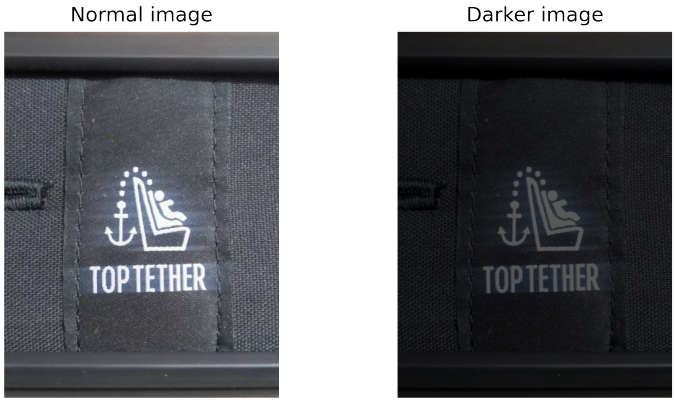


Figure 2.2: Phone's images : normal to darker

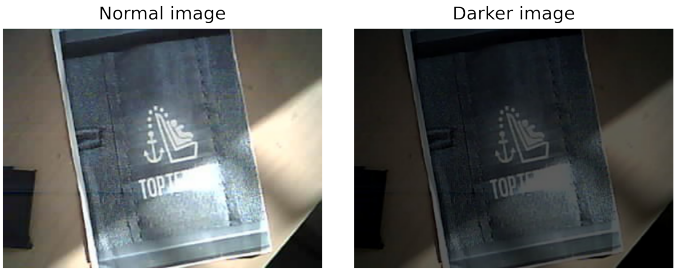


Figure 2.3: esp32-cam on printed images : normal to darker

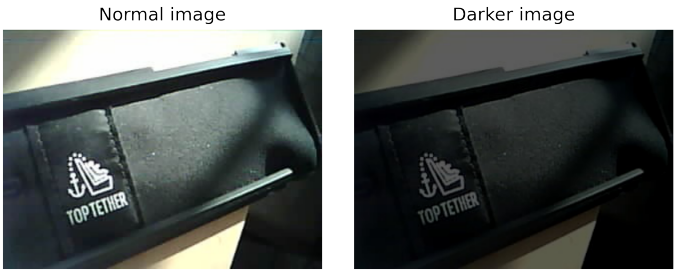


Figure 2.4: esp32-cam on the actual product : normal to darker

```
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 [9] using this equation:

$$output\_Pixel = \alpha * input1\_Pixel + \beta * input2\_pixel + \gamma[9] \tag{2.1}$$

where:

- output_Pixel = output pixel value
- α = weight given to the first input image (input1_pixel)
- input1_Pixel = first source array
- β = weight given to the second input image (input2_pixel)
- input2_Pixel = second source array
- γ = optional scalar value added to every output pixel value

0.2 reflection

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.

0.3 grayscaling

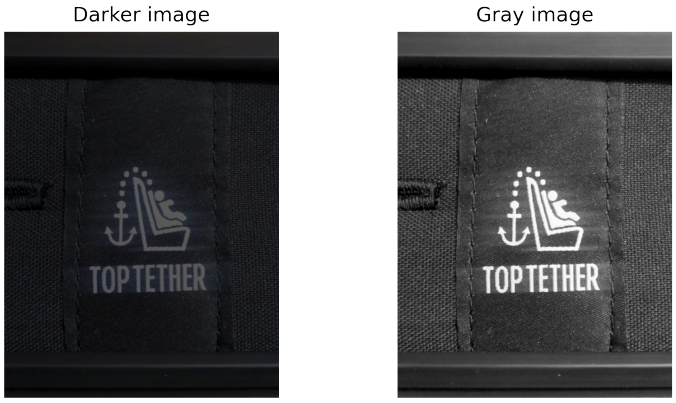


Figure 2.5: Phone's images :darker to gray

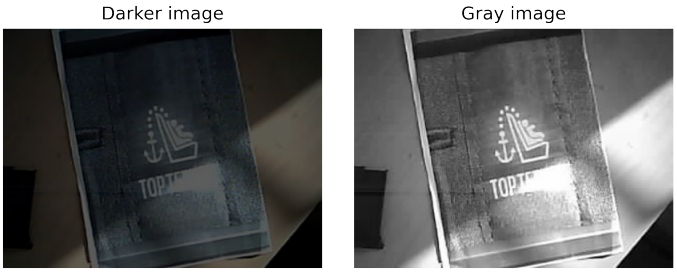


Figure 2.6: esp32-cam on printed images : darker to gray

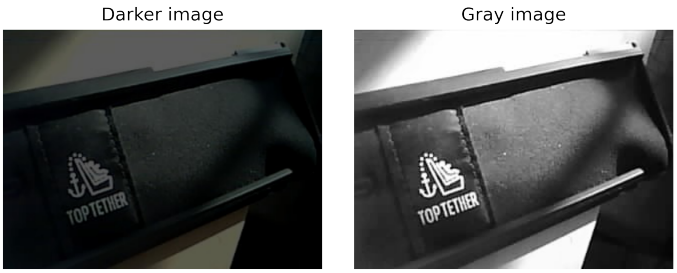


Figure 2.7: esp32-cam on the actual product : darker to gray

```
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:[9]

$$result = 0.299R + 0.587G + 0.114B[9] \tag{2.2}$$

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.

0.4 reflection

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.

0.5 Image thresholding

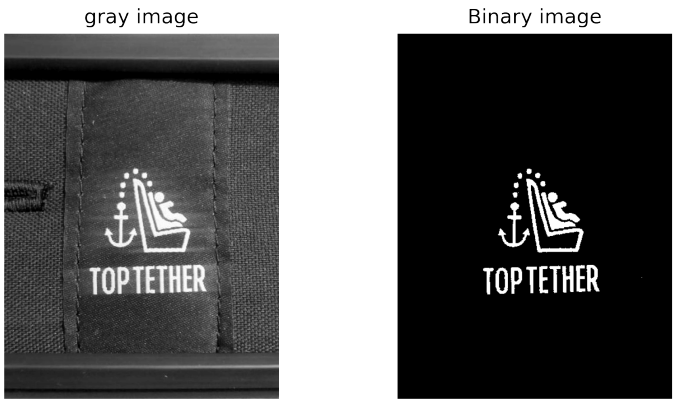


Figure 2.8: Phone's images :Gray to binary

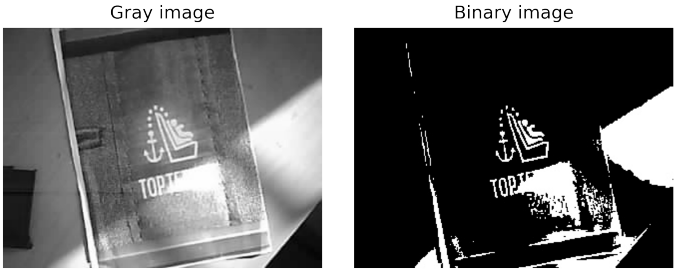


Figure 2.9: esp32-cam on printed images : Gray to binary

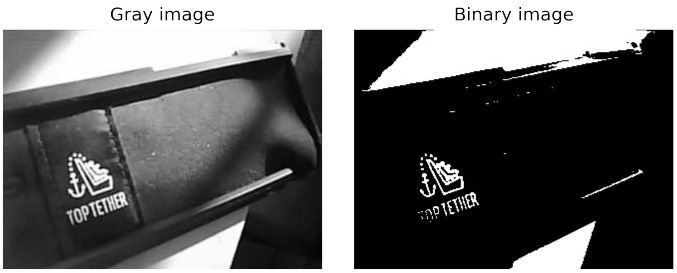


Figure 2.10: esp32-cam on the actual product :Gray to binary

```
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 [9] using this equation:

$$dst(x,y) = \begin{cases} maxVal & \text{if } src(x,y) > T \\ 0 & \text{otherwise} \end{cases} \text{ [9]} \tag{2.3}$$

- $dst(x,y)$: 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

0.6 reflection

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.

0.7 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

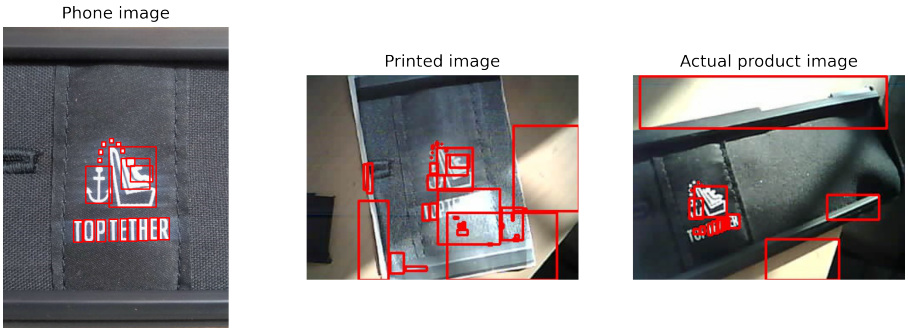


Figure 2.11: prediction results

```
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` [8], 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 ϵ. 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 ϵ, 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.

Conclusion

In conclusion, the first sprint 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 sprint 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 the first sprint 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

Sprint Two: Raspberry pi 4 and ESP32 Cam Integration for Logo Detection

Introduction

I Specification of requirements

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

1 actors identification

2 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 indicating whether the logo is present and if it is flipped or not.

3 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.

- **Security:** The system should have appropriate security measures in place to prevent unauthorized access to the servers and data.
- **Scalability:** The system should be able to handle multiple simultaneous connections and requests from clients without compromising its performance.
- **Maintainability:** The system should be easy to maintain and update, with clear and well-documented code and configuration.
- **Compatibility:** The system should be compatible with a wide range of devices and platforms.
- **Usability:** The system should be easy to use and understand for both technical and non-technical users.

II Modeling languages diagrams

1 SysMl

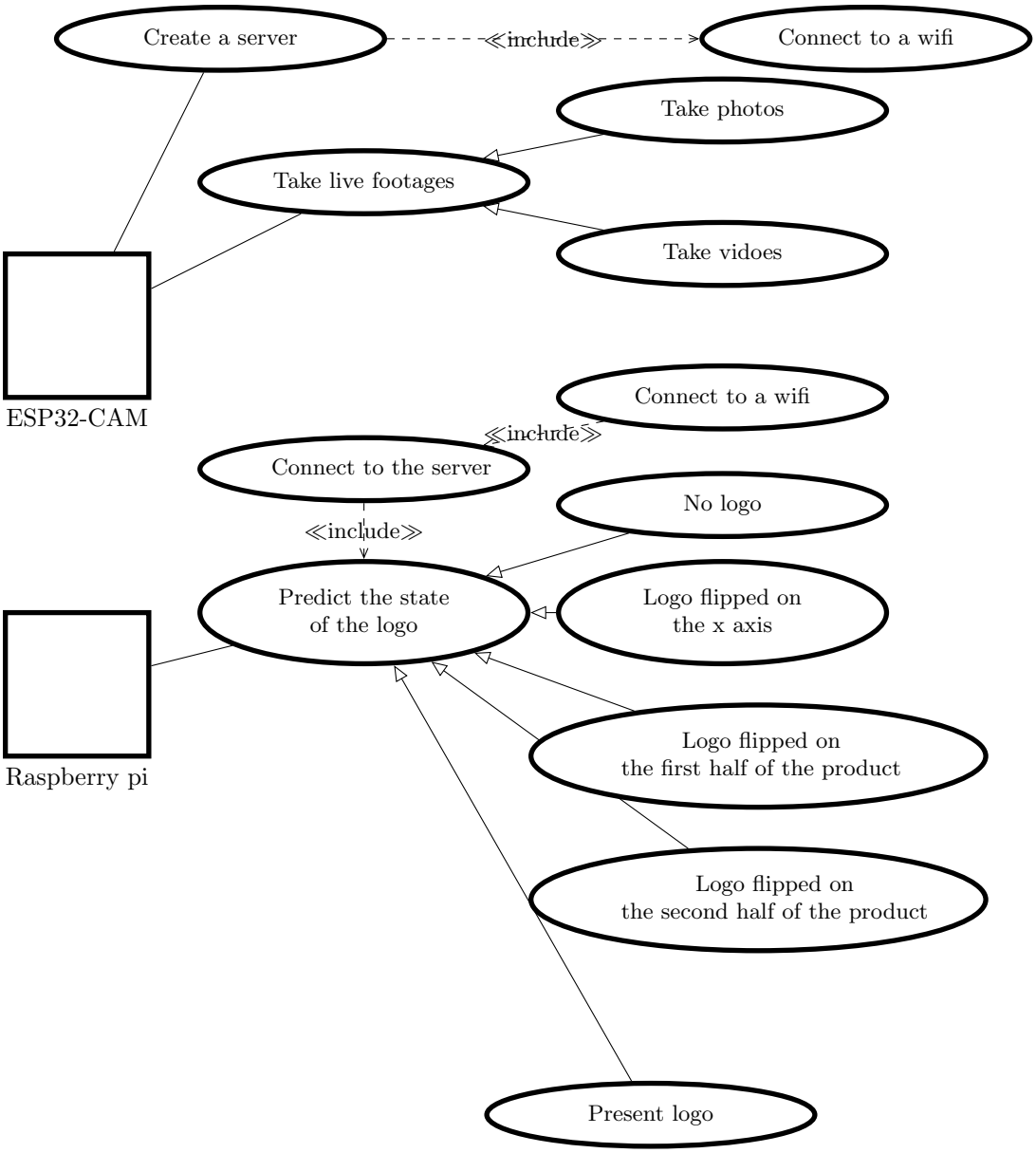


Figure 3.1: Use case Diagram for sprint version 1

III Project component

1 Hardware and Software environment

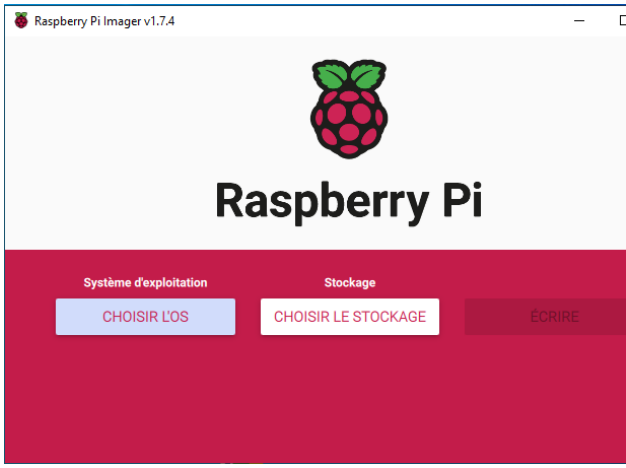
For a comprehensive understanding of the hardware and software environment components used in this study, I recommend referring to the annex chapter **??** of this report. The annex contains detailed information on the system specifications and configurations, including hardware components such as ESP32-cam, 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.

IV Workflow

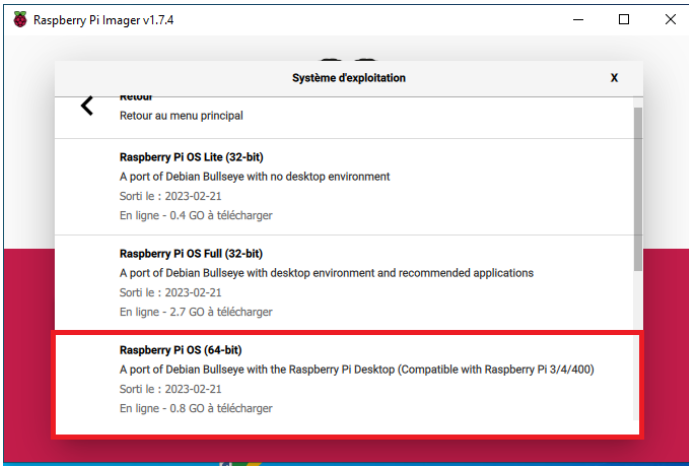
0.1 Setting up the rapsberry pi

- setting up the OS

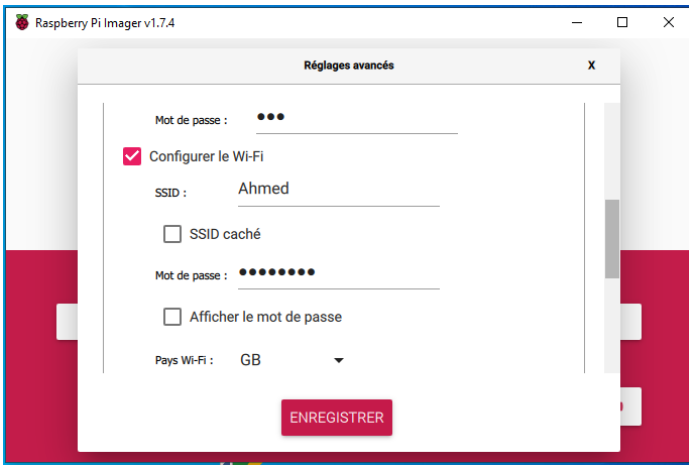
  – download the Raspberry Pi Imager from the official Raspberry Pi website.



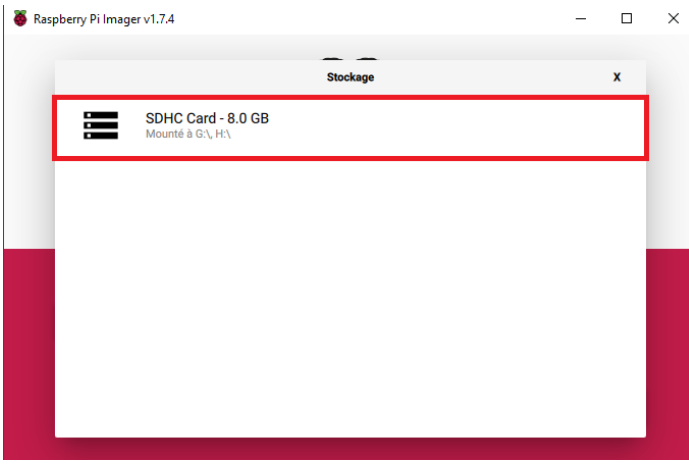
  – choose the raspberry pi operating system (64-bit)



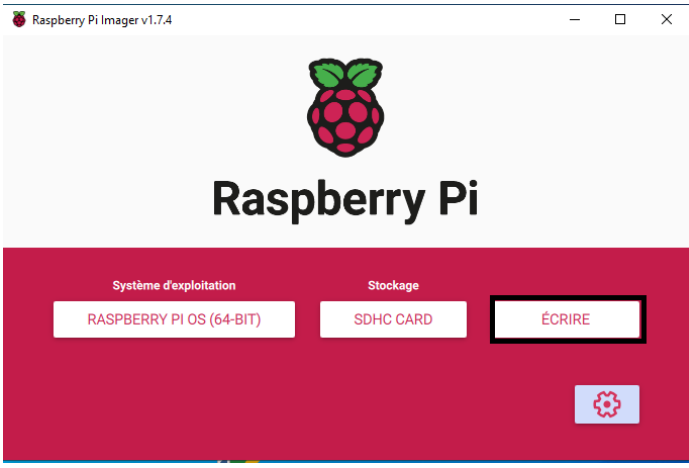
– Configure the parameters of the operating system: set a hostname,enable SSH,set username and password,configure wireless lan and set local settings



_ 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.



- setting up the VNC

– Establishing connection with puttty

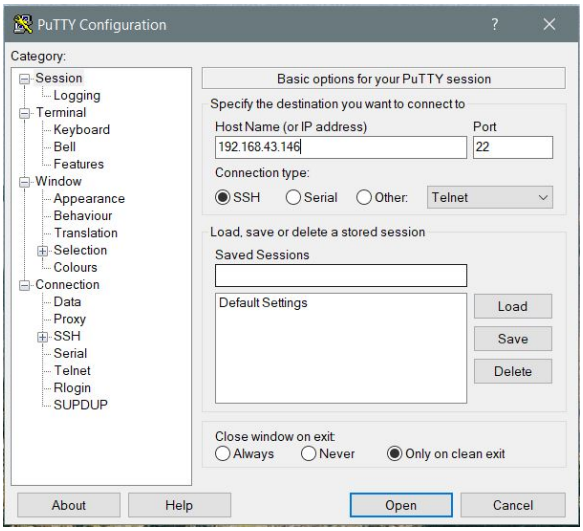


Figure 3.2: putty interface

– 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

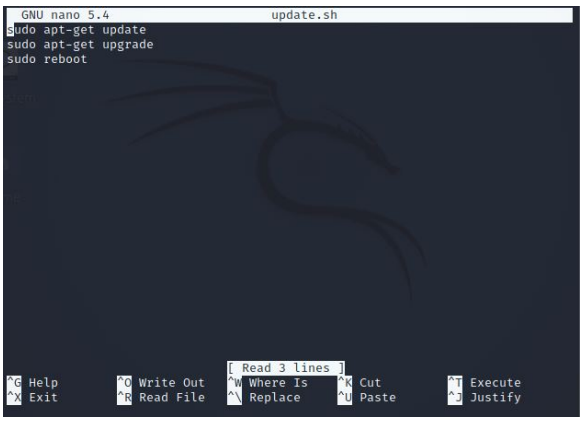


Figure 3.3: Update.sh file

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
```

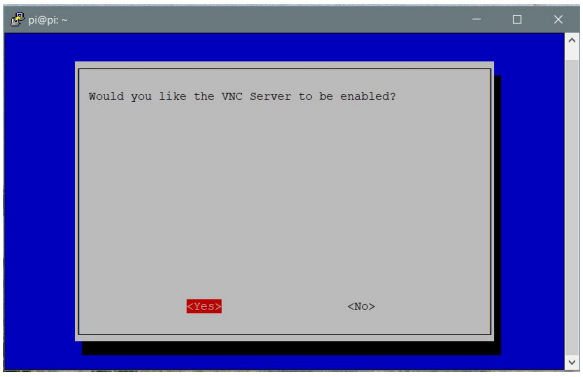


Figure 3.4: Enable vnc menu

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

- Prepare working environment

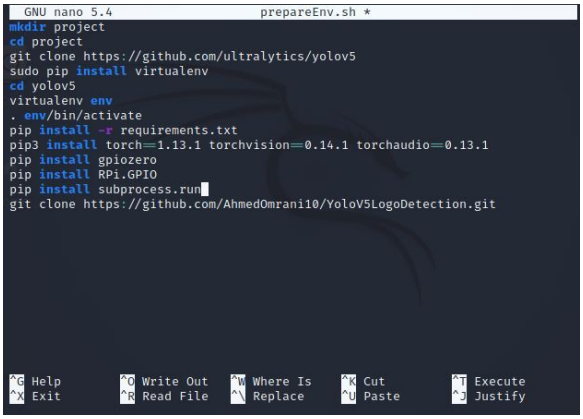


Figure 3.5: prepareEnv.sh file

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

0.2 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.[18]
  - 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.[18]
  - 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.[20]
  - 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.[20]

In Table **??** and the firgure 3.6, we compare the performance of different YOLOv5 models.

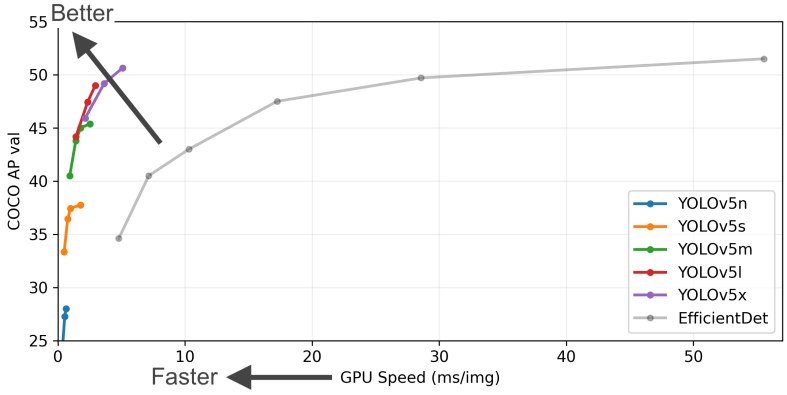


Figure 3.6: Comparison between yolov5 family and Efficientdet for the COCO dataset in term of speed and accuracy

Comparison of YOLOv5 Models [21]						
Model	**Size (pixels)**	**mAPval@50-95**	**mAPval@50**	**Speed CPU b1 (ms)**	**Speed V100 b1 (ms)**	**Speed V1**
YOLOv5n	640	28.0	45.7	45	6.3	
YOLOv5s	640	37.4	56.8	98	6.4	
YOLOv5m	640	45.4	64.1	224	8.2	
YOLOv5l	640	49.0	67.3	430	10.1	
YOLOv5x	640	50.7	68.9	766	12.1	

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".[21]
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".[21]
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".[21]

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 this table **??** 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

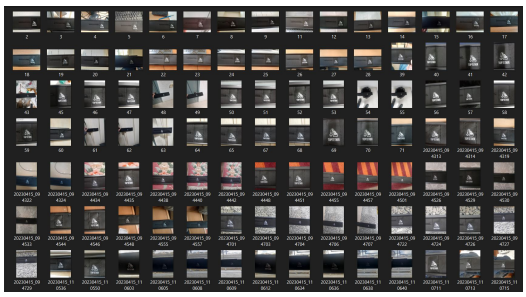


Figure 3.7: Normal dataset

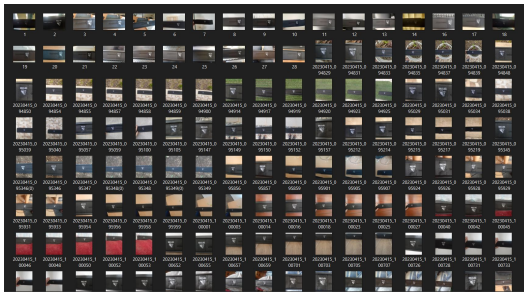


Figure 3.8: Inverted dataset

To train our Yolov5s model, we first gathered a dataset of 726 images containing logos. We split the dataset into two classes: Normal and Inverted. The Normal class 3.7 contained 363 images of logos in their standard orientation, while the Inverted class 3.8 contained 363 images of logos that had been rotated 180 degrees.

- Image labeling

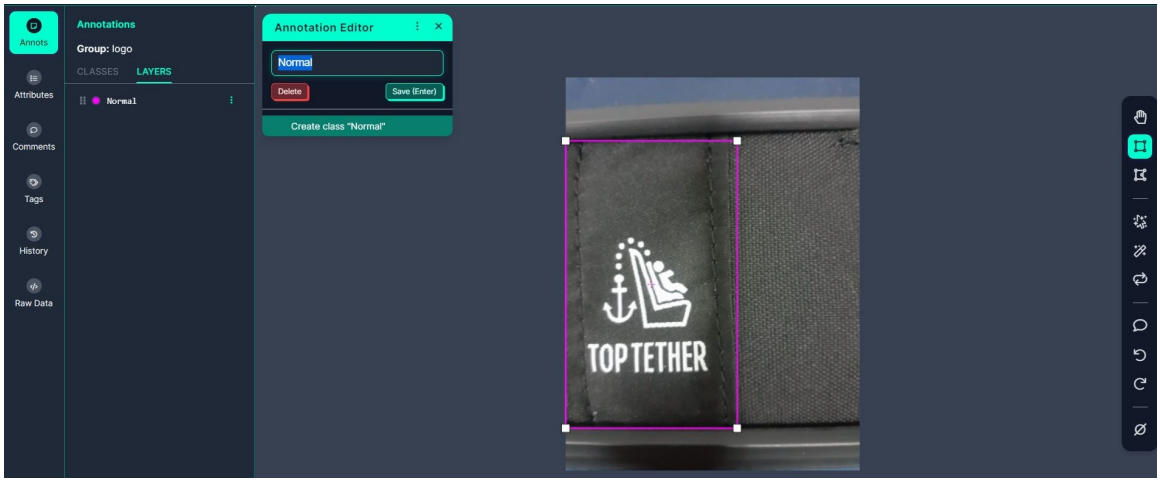


Figure 3.9: Labeling an image

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

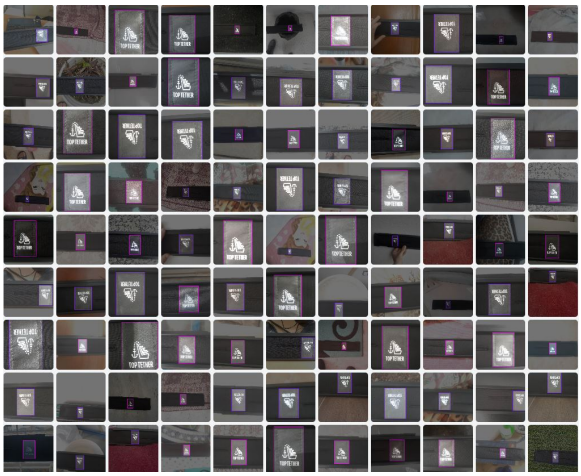


Figure 3.10: Augmented dataset

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.

```
  %%writetemplate /content/yolov5/models/custom_yolov5s.yaml

# parameters
nc: {num_classes} # number of classes
depth_multiple: 0.33 # model depth multiple
width_multiple: 0.50 # layer channel multiple
anchors:
  - [10,13, 16,30, 33,23] # P3/8
  - [30,61, 62,45, 59,119] # P4/16
  - [116,90, 156,198, 373,326] # P5/32

# YOLOv5 v6.0 backbone
backbone:
  # [from, number, module, args]
  [[-1, 1, Conv, [64, 6, 2, 2]], # 0-P1/2
   [-1, 1, Conv, [128, 3, 2]], # 1-P2/4
   [-1, 3, C3, [128]],
   [-1, 1, Conv, [256, 3, 2]], # 3-P3/8
   [-1, 6, C3, [256]],
   [-1, 1, Conv, [512, 3, 2]], # 5-P4/16
   [-1, 9, C3, [512]],
   [-1, 1, Conv, [1024, 3, 2]], # 7-P5/32
   [-1, 3, C3, [1024]],
   [-1, 1, SPPF, [1024, 5]], # 9
  ]

# YOLOv5 v6.0 head
head:
  [[-1, 1, Conv, [512, 1, 1]],
   [-1, 1, nn.Upsample, [None, 2, 'nearest']],
   [[-1, 6], 1, Concat, [1]], # cat backbone P4
   [-1, 3, C3, [512, False]], # 13

   [-1, 1, Conv, [256, 1, 1]],
   [-1, 1, nn.Upsample, [None, 2, 'nearest']],
   [[-1, 4], 1, Concat, [1]], # cat backbone P3
   [-1, 3, C3, [256, False]], # 17 (P3/8-small)

   [-1, 1, Conv, [256, 3, 2]],
   [[-1, 14], 1, Concat, [1]], # cat head P4
   [-1, 3, C3, [512, False]], # 20 (P4/16-medium)

   [-1, 1, Conv, [512, 3, 2]],
   [[-1, 10], 1, Concat, [1]], # cat head P5
   [-1, 3, C3, [1024, False]], # 23 (P5/32-large)

   [[17, 20, 23], 1, Detect, [nc, anchors]], # Detect(P3, P4, P5)
  ]
```

In this case and as you can see from the code above , 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 validate 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 and Validation 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.

Training and 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.

Training and Validation 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.
  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.

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 or inverted position and if it doesn't we can coclude that there is no logo.

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 by assigning xmin, ymin, xmax, ymax to the variables row[0], row[1], row[2], and row[3], respectively.

```
xmin, ymin, xmax, ymax = row[0], row[1],row[2],row[3]
```

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

Chapter 4

Sprint 2: Stabilizing the System

Chapter 5

Sprint 4: PCB Board and Case Design

Chapter 6

Annex

I Project component

1 Hardware environment

1.1 ESP32-CAM

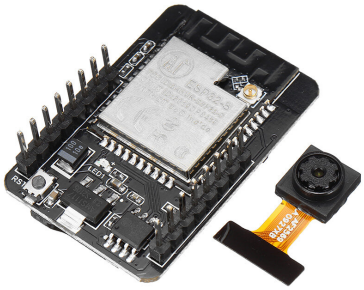


Figure 6.1: ESP32-CAM

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

Table 6.1: ESP32-CAM characteristics [10]

1.2 PC

Figure 6.2: lenovo ideapad gaming 3

2 Software environment

2.1 Software

- Arduino IDE

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

Table 6.2: Specifications of the PC

Figure 6.3: 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 6.4: 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

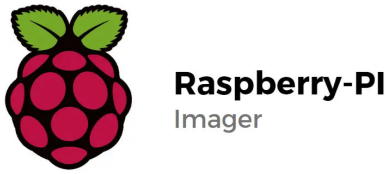


Figure 6.5: Raspberry pi imager logo

Raspberry Pi Imager is a free and open-source software application developed by the Raspberry Pi Foundation.[12] 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.[12]

- PuTTY

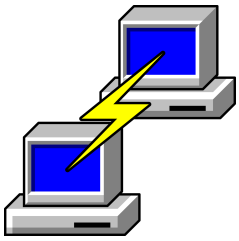


Figure 6.6: puttyLogo

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.[13]

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.[13]

- VNC

Figure 6.7: 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.[14]

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.[14]

- Geany

Figure 6.8: 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.[15]

- Google Colab

Figure 6.9: 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. .[16]

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..[16]

2.2 programming languages

- C++

Figure 6.10: 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

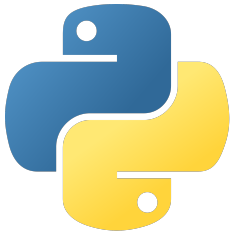


Figure 6.11: Python logo

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] M. A. Awad, 2015. *A Comparison between Agile and Traditional Software Development Methodologies* .

[2] Pete Deemer, Gabrielle Benefield, Craig Larman, Bas Vodde, 2010. *THE SCRUM PRIMER.*

[3] Hans-Erik Eriksson, Magnus Penker, Brian Lyons , 2003. *UML 2 toolkit.*

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

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

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

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

[12] Raspberry Pi Software,(n.d.). *https://www.raspberrypi.com/software/*

[13] PuTTY. (n.d.). *https://www.putty.org/*

[14] RealVNC. (n.d.). *https://discover.realvnc.com/what-is-vnc-remote-access-technology*

[15] Geany. (n.d.). *https://www.geany.org/*

[16] Google Research . (n.d.). *https://research.google.com/colaboratory/faq.html*

[17] 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.*

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

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

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

[21] ultralytics, 2023. *yolov5 github repository : https://github.com/ultralytics/yolov5* .