

Palestine Polytechnic University



College of IT & Computer Engineering

**This project was submitted as a requirement to obtain the bachelor's
degree in computer systems engineering**

**Smart Cashier System using image processing and thermal
packaging**

Supervisor:

Dr. Ayman Wazwaz

Team Members:

Jaafar Alrjoob

Asem Abu Hadwan

Abstract

Our project introduces the design and implementation of the 'Smart Cashier System,' an innovative solution revolutionizing the fruit purchasing experience. Powered by a Raspberry Pi 4 micro-computer, the system integrates an ESP32 microcontroller for sensor interaction, a camera for precise fruit recognition, and an intuitive touchscreen interface.

The essence of the Smart Cashier System lies in its ability to enhance fruit purchasing through automation and personalization. Customers can effortlessly select a fruit swiftly recognized by the system's camera, with its weight accurately converted into a price displayed on the user-friendly touchscreen.

This experience is made possible by the integration of a YOLOv8 model, renowned for its exceptional fruit identification accuracy. The system incorporates a load cell to meticulously measure the fruit's weight, ensuring transparent transactions by automatically computing the price based on predefined formulas.

Furthermore, to expedite the checkout process, the system generates QR codes embedded with product information and pricing details. Customers can effortlessly scan these codes and proceed with payment, enhancing efficiency and convenience in fruit selection and purchase."

Acknowledgement

We would like to express our sincere gratitude to all those who have supported us throughout this project. We would like to give special thanks to our supervisor Dr. Ayman Wazwaz, who provided valuable guidance and support, and to our colleagues who have shared their knowledge and expertise. We would also like to thank our families and friends for their unwavering support and encouragement throughout the process. Lastly, we would like to acknowledge the hard work and dedication of all those who contributed to the success of this project. Thank you all.

Table of Contents

ABSTRACT	2
ACKNOWLEDGEMENT	3
TABLE OF CONTENTS	4
CHAPTER 1	6
INTRODUCTION	6
1.1 <i>Overview</i>	6
1.2 <i>Motivation</i>	6
1.3 <i>Objectives</i>	6
1.4 <i>Challenges</i>	7
1.6 <i>Requirements</i>	7
1.6.1 <i>Functional requirements</i>	7
1.6.2 <i>Non-functional requirements</i>	7
1.7 <i>Project Details</i>	8
1.8 <i>General block diagram</i>	8
CHAPTER 2	9
BACKGROUND AND LITERATURE REVIEW	9
2.1 <i>Introduction</i>	9
2.2 <i>Theoretical Background</i>	9
2.2.1 <i>Image processing</i>	9
2.2.2 <i>Thermal packaging</i>	9
2.2.3 <i>QRcode generating & printing</i>	10
2.3 <i>Literature review</i>	10
2.3.1 <i>Fruit Recognition on a Raspberry Pi</i>	10
2.3.2 <i>Smart POS System Cash Register with Weighing Checkout Scale:</i>	11
2.4 <i>Hardware Components</i>	13
CHAPTER 3	21
DESIGN	21
3.1 <i>Introduction</i>	21
3.2 <i>Modeling Diagrams</i>	21
3.2.1 <i>Block Diagram</i>	21
3.2.2 <i>Flow Charts</i>	21
3.3 <i>Software design</i>	24
3.3.1 <i>Object Detection with Yolov8</i>	24
3.3.1.1 <i>Advantages of YOLOv8</i>	24
3.3.2 <i>QR code generating</i>	24
3.3.2.1 <i>Advantages of QR Codes</i>	25
3.4 <i>Schematic Design</i>	25
3.5 <i>Exterior design</i>	27
CHAPTER 4	28
IMPLEMENTATION AND RESULTS	28
4.1 <i>Overview</i>	28
4.2 <i>Hardware Implementation</i>	28
4.2.1 <i>Raspberry PI Circuit</i>	30
4.2.2 <i>Esp32 Circuit</i>	31
4.3 <i>Software Implementation</i>	32
4.3.1 <i>Raspberry Pi Setup</i>	32

4.3.1.1 Operating System	33
4.3.1.2 VNC Viewer	33
4.3.1.3 Raspberry pi Camera setup	33
4.3.1.4 Touchscreen Calibration	36
4.3.1.5 Installing Libraries	37
4.3.2 Image Processing	38
4.3.2.1 Dataset Preparation	38
4.3.2.2 Train the Model	39
4.3.2.3 Deploying the model	40
4.3.3 Esp32 Setup	40
4.3.4 System User Interface	40
4.3.5 Coding	41
CHAPTER 5	47
TESTING AND VALIDATION	47
5.1 <i>Overview</i>	47
5.2 <i>Hardware Testing</i>	47
5.2.1 Weighing System Testing (Load cell testing)	47
5.2.2 Linear Actuator and Impulse Sealer	48
5.2.3 Touch Screen Testing	49
5.3 <i>Software Testing</i>	49
5.3.1 Image Process Testing	49
5.3.1.1 Evaluate The Model	50
5.3.1.2 Test The Model	52
5.3.2 QR code Testing	53
5.3.3 UI Testing and Functional Testing	53
CHAPTER 6	55
CONCLUSION AND FUTURE WORK	55
6.1 <i>Summary of Deliverables</i>	55
6.2 <i>Challenges and Overcoming Strategies</i>	55
6.3 <i>Interesting Decisions</i>	56
6.4 <i>Future Work</i>	56
LIST OF FIGURES	58
LIST OF TABLES	59

Chapter 1

Introduction

1.1 Overview

The Smart Cash project aims to create a new device that will change the way we buy and sell fruits and vegetables. This device uses modern technology, like image processing and detection, to identify and weigh the fruits and vegetables. Then, it displays their information on a screen and creates an invoice with a barcode. The device also has a thermal packaging system to make sure the products are well-packaged.

1.2 Motivation

The reason behind the Smart Cash project is to solve the problems faced by markets and grocery stores in weighing and pricing fruits and vegetables. The Smart Cash device will eliminate the need for traditional weighing scales and manual price calculation. The device is easy to use and provides accurate and reliable information to both buyers and sellers, making it a valuable addition to any market or grocery store.

1.3 Objectives

- Develop a smart cashier system that uses image processing to accurately identify different types of fruits and vegetables.
- Implement a system that can quickly and accurately calculate the weight and price of several types of fruits and vegetables.
- Display the information about each item, including its name, weight, and price, on a screen.
- Integrate a thermal packaging system to ensure that the bag of fruits or vegetables will not be tampered with after being closed and after the total price is generated.
- Provide a self-cashier by printing a QR code on a paper invoice containing the price of the final product.

1.4 Challenges

- Difficulty in identifying different types of fruits and vegetables through image processing techniques (due to brightness for example), our system solves this by making the image processing environment isolated to ensure that one lighting and one background is obtained when processing images of fruits.
- High production costs expenses that could make the device unaffordable for smaller markets and grocery stores, we solve this problem by using alternative components, and we choose the cheapest of these components that serve the purpose.
- Without proper packaging, the products may spoil or get tampered, leading to more inconvenience and losses for buyer and seller, we solve this problem by making the process of packaging and pasting the barcode out of sight of the user, and this solution also prevents the buyer from tampering with the barcode after it is pasted.

1.5 Problem Statement

- Using traditional weighing scales and manual price calculation takes a lot of time and mistakes can happen easily, which creates problems for both buyers and sellers.
- The need for the human workers in grocery stores or in the fruit's departments in malls.

1.6 Requirements

There are several functional and non-functional requirements of the system:

1.6.1 Functional requirements

- The system must be able to identify different types of fruits and vegetables using image processing techniques.
- The system must be able to run quickly and calculate the weight and price of several types of fruits and vegetables.
- The system must display the information about each item on a screen.
- The system must print the barcode on a paper invoice, containing the price of the final product.

1.6.2 Non-functional requirements

- The system must be user-friendly and easy to use.
- The system must be reliable and accurate in its calculations.

- The system must be secure to prevent tampering with the final price or the products being sold.

1.7 Project Details

The system consists of several components, including a microcontroller, an LCD touch screen, a camera, and sensors for weighing the fruits and vegetables. The device utilizes image processing and detection techniques to accurately identify the type of product being placed on it. It then calculates the weight and price of the product and displays the information on the LCD screen.

The thermal packaging system is an additional feature of the device that ensures that the bag of fruits or vegetables is properly sealed and not tampered with after the total price is generated. The device also generates a barcode on the printed invoice containing the price of the final product.

1.8 General block diagram

We designed a simplified block diagram of our system. It describes how the system components interact with the each other. The system runs on a microcomputer where the calculations and machine learning models exist. The microcomputer receives data from the sensors, receives image from the camera and doing its calculations then print the results on the screen, and generate the barcode to print it later, finally the system will move to thermal packaging process. Figure 1 shows the general block diagram.

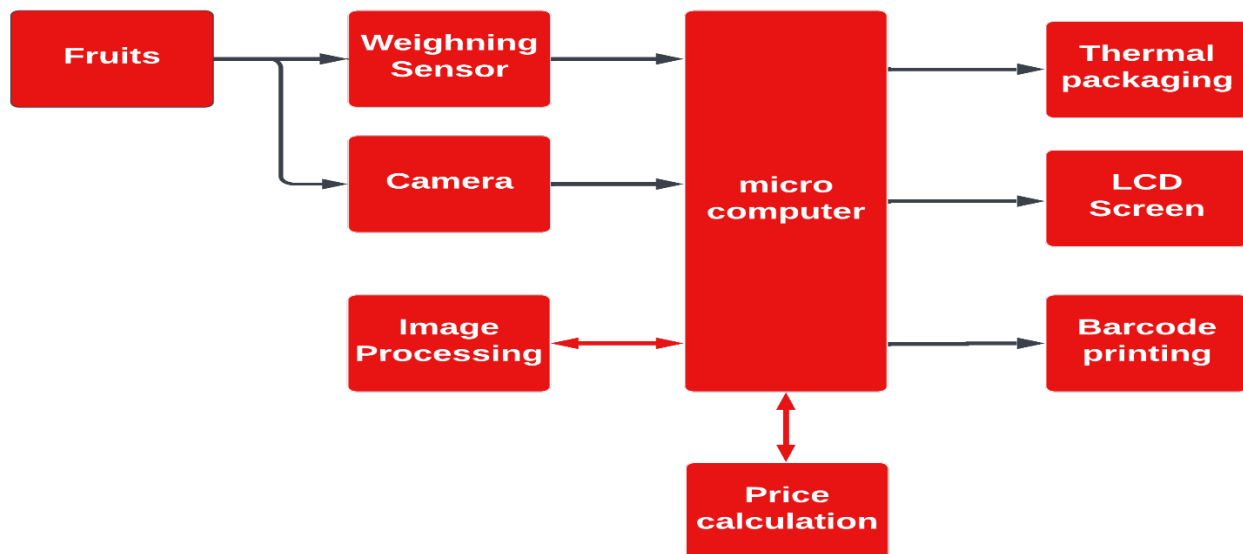


Figure 1 General Block Diagram

Chapter 2

Background and Literature Review

2.1 Introduction

The Smart Cash device is a technology that makes it easier to buy and sell fruits and vegetables. It uses image processing and detection to identify and weigh the product, and displays the information on a screen. This means that traditional weighing scales and manual price calculation are no longer necessary. To create a functional Smart Cash device, this project needs to review the current researches on image processing and thermal packaging, and reviewing the similar previous projects. This chapter will provide a detailed review of this research, which is important for developing the Smart Cash device.

2.2 Theoretical Background

The following sections include the most important topics involved in the project.

2.2.1 Image processing

Image processing is a way to use computers to understand and change pictures. It can be used for many things, like looking at medical images or watching for bad behavior on security cameras.

For the fruit and vegetable recognition, image processing will be very important. First, a picture is taken of the fruit or vegetable. Then, the computer uses different tools to understand the picture like color and size. One of the most important tools is called segmentation, which helps the computer separate the fruit or vegetable from the background. [1]

2.2.2 Thermal packaging

Thermal packaging refers to the use of specialized packaging materials and techniques to regulate the temperature of perishable goods, such as fruits and vegetables, during transportation and storage. The aim is to maintain the optimal temperature range for the produce to reduce spoilage, maintain freshness, and extend shelf life. [2]

In the project, thermal packaging will play an essential role in preparing the fruits and vegetables for barcode affixing. After the produce is sorted and packaged, it will be placed in thermal

packaging to ensure that the temperature is maintained at the appropriate level until it reaches the point of sale. This helps to ensure that the produce is fresh and has a longer shelf life.

2.2.3 QRcode generating & printing

QRcode printing is a way to create a pattern of bars and spaces that can be read by a scanner. This pattern contains information about the product, such as the price or a unique identification number. When a scanner reads the QRcode, it can quickly and accurately identify the product and retrieve the information stored in the QRcode. In the case of packed fruits and vegetables, a QRcode can be used to store the final price of the product. [3]

2.3 Literature review

The identification of fruits and vegetables is implemented in various fields, with the most common being the retail business and agriculture. In the retail business, identification is typically done manually by a cashier or through self-service systems in a store.

Through our search we find several relevant literatures and we mentioned some of them below:

2.3.1 Fruit Recognition on a Raspberry Pi

The project involves developing a fruit recognition system using a Raspberry Pi, a camera, and a neural network. The camera observes a clean table, and as soon as a user puts a fruit onto the table the user can hit a button on a shield attached to the Raspberry Pi. The button triggers the camera to take an image, which is then fed into a trained neural network for image categorization. The category is then spoken out loud using a speech synthesizer, as shown in figure 2. [4]

Features:

- The image is fed into a trained neural network for image categorization.
- Uses a multi-categorical neural network to categorize fruit images.
- Images are cropped to the size of the fruit for standardization.
- Bright backgrounds are replaced with darker backgrounds for more realistic images.
- The neural network model consists of four convolutional layers, with the number of filters increasing with each layer.

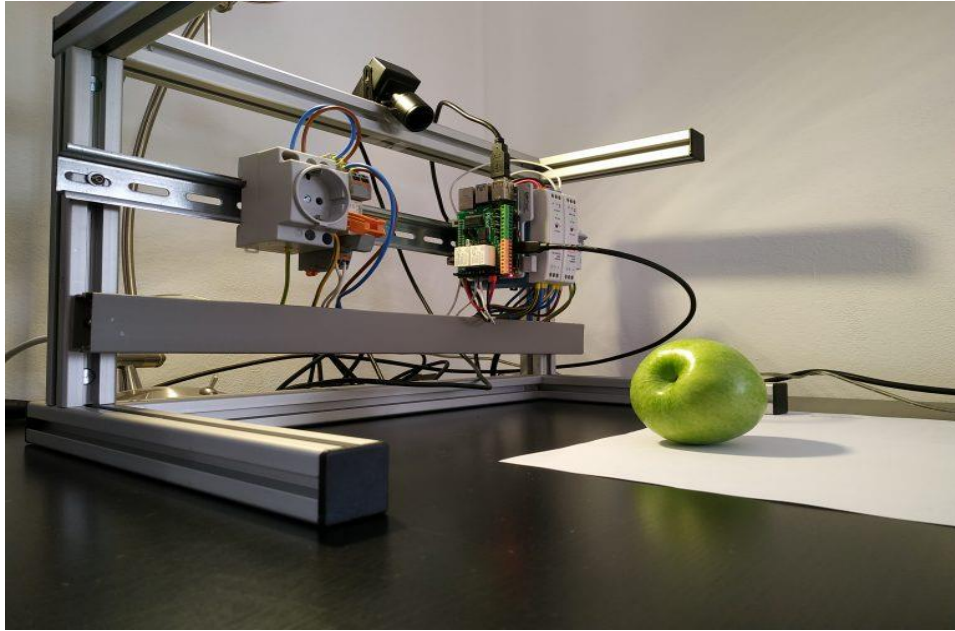


Figure 2 Fruit Recognition on a Raspberry Pi [5]

Limitations:

- The fruit images used for training may not be representative of real-life variations in fruit appearance.
- The bright backgrounds in the original images may not accurately reflect real-life backgrounds.
- The model's performance may be affected by variations in lighting and other environmental factors.
- The project only recognition fruits without doing any action, and cannot be consider as a commercial product.
- The project does not integrate thermal packaging and barcode printing.

2.3.2 Smart POS System Cash Register with Weighing Checkout Scale:

The A4-116-TM AI smart cash register scale is an innovative solution for retail businesses in need of a high-integration commercial equipment. With its perfect design integrating POS machine and electronic scale, this product offers a precise weighing range of 0.01-30LB that can be customized to 0.01-15KG. It comes with a dual LCD screen, allowing for clear interface and the second display can be used for broadcasting advertisement. This product runs on the Windows 10 system, making it easy to connect to a wireless network and install the other software needed. With its strong compatibility with hardware and software, it can be used with peripheral products such as thermal printers, scanners, card readers, keyboards, and cash boxes. Overall, it's an intelligent all-in-one

design with a user-friendly software making it an ideal solution for fruit stores, snack stores supermarkets, bakeries, vegetable stores, delicatessen, seafood stores, or any other retail businesses in need of weighing checkout together. [5] figure 3 shows the product.



Figure 3 Smart POS System Cash Register [6]

Features:

- Dual LCD screens for clear interface and advertising display
- Integration of POS machine and electronic scale
- Windows system for data transmission between front and back office
- Built-in thermal receipt printer and label printer
- Strong compatibility with hardware and software, can connect to wireless network and install other software
- Applicable for fruit stores, snack stores, supermarkets, bakeries, vegetable stores, delicatessens seafood stores, and other retail businesses

Limitations:

- Does not recognition the fruits & vegetables automatically, the user have to choose the type of fruits manually.

- May require additional software or hardware for specific business needs.
- Precise weighing may not be suitable for all types of products.
- May require professional installation and setup.
- Does not integrate thermal packaging.
- The system is sold as a one part and you cannot buy the scale separately.
- The product price is very expensive: 1,549.90 \$

2.4 Hardware Components

In order to build this project, we used many electronic parts and connected them together to get the required functionality.

Now we summarize the list of used electronic parts as follows:

1- Load cell sensor:

In order to obtain the weight for vegetables and fruits, we chose to build the scale from scratch using the load cell sensor.

A load cell is a type of sensor that is used to measure force or weight in a wide range of applications. Load cells are commonly used in industrial, commercial, and scientific contexts where accurate and reliable force measurements are required. [6] The sensor datasheet is shown in table 1.

Depending on the project requirements we decide to choose 5kg load cell sensor because it meets the requirements of our project, since the weight of fruits or vegetables will not be more than a 5kg as represented in figure 4.

Table 1 Load Cell sensor data sheet

Specifications	
Mechanical	
Housing Material	Aluminum Alloy
Capacity	1/2/3/5 kg
Dimensions	Lx12.7x12.7 mm
Mounting Holes	M5 (Screw Size)
Cable Length	210 mm
Cable Size	30 AWG (0.2mm)
Cable - no. of leads	4
Electrical	
Precision	0.05%
Rated Output	1.0±0.15 mV/V
Temperature Effect on Zero (per 10°C)	0.02% FS
Temperature Effect on Span (per 10°C)	0.05% FS Zero
Balance	±1.5% FS
Insulation Resistance (Under 50VDC)	≥2000 MOhm
Excitation Voltage	5 VDC
Compensated Temperature Range	-10 to ~+40°C
Operating Temperature Range	-21 to ~+40°C
Safe Overload	120% Capacity
Ultimate Overload	150% Capacity

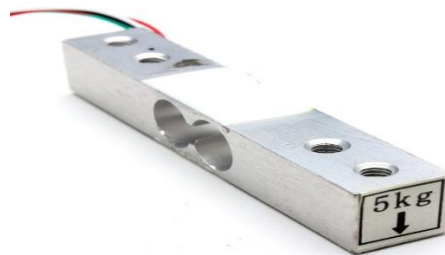


Figure 4 Load cell sensor [8]

2- LCD Module:

In order to show information about vegetables and fruits, their weight and price, our project needs a screen that performs this purpose, and compatible with the Raspberry Pi. While searching for a screen that meets the requirements of the project, we found several options that fit the requirements. Consider that the primary goal of the screen is to display information only and the size and quality of the display are not important factors, then a smaller and more affordable LCD module may be a suitable option. Based on this criterion, a 7" Touchscreen Display could be good option. And more affordable compared to the other LCD modules listed in the table. Figure 5 shows the LCD module.



Figure 5 LCD Module [9]

3- ESP32:

During to our project need of a Microcontroller, we made a search of available micro-controllers and find that the most common used controllers are Arduino nano and ESP32 and compare between them as shown in Table 2.

The result was clearly shown that ESP32 is a better and more powerful microcontroller board than Arduino, and because the price is cheaper than other competitors, we decide to use ESP32. Figure 6 shows the esp32.

Table 2 Comparison between microcontrollers [7]

Microcontroller	ESP32	Arduino Nano
CPU Clock Frequency	Up to 240 MHz	Up to 16 MHz
RAM	520 KB	2 KB
Flash Memory	4 MB	32 KB
Connectivity	Wi-Fi, Bluetooth, Ethernet, CAN bus	No built-in connectivity
Analog Inputs	18	8
Digital I/O Pins	36	14
PWM Pins	16	6
ADC Resolution	12-bit	10-bit
Operating Voltage	3.3V	5V
Power Consumption	Low power consumption	Higher power consumption
Programming Language	C/C++, MicroPython, JavaScript, Lua	C/C++
Price	20\$	15\$



Figure 6 Esp32

4- Raspberry Pi 4:

According to our project's need for a micro-computer in order to perform the image processing and the calculations needed, during our search for a suitable micro-computer, we choose to use the Raspberry Pi 4.

The Raspberry Pi4 is a single-board computer developed by the Raspberry Pi Foundation. It is the latest model in the Raspberry Pi series and offers several upgrades over its predecessors, including improved processing power, better multimedia support and faster connectivity. The Raspberry Pi 4 comes with a quad-core 64-bit ARM Cortex-A72 CPU running at 1.5GHz, up to 8GB of RAM, and support for dual 4K displays. It also features Gigabit Ethernet, dual-band 802.11ac wireless, and Bluetooth 5.0. [8] Figure 7 shows the raspberry Pi 4.



Figure 7 Raspberry Pi 4

maybe it is not the most powerful micro-computer, but it is a versatile and widely-used device with a large community and a wide range of compatible software and hardware, due to its low price Compared to other microcomputers, and since it covers the project's needs in terms of RAM capacity, and it is available in our region and easy to obtain and purchase we decide to use Raspberry Pi4.

5- Camera Module:

For image processing applications involving fruits and vegetables, resolution and field of view are likely to be important factors. Higher resolution can provide more detailed images, while a wider field of view can capture a larger area and potentially more objects in a single image. Based on these criteria, the Raspberry Pi Camera Module V2 could be good option for image processing.

The Raspberry Pi Camera v2 is a high quality 8-megapixel Sony IMX219 image sensor custom designed add-on board for Raspberry Pi, featuring a fixed focus lens. It's capable of 3280 x 2464-pixel static images, and also supports 1080p30, 720p60 and 640x480p60/90 video. [9] Figure 8 shows the Raspberry Pi Camera v2 camera module.



Figure 8 Raspberry Pi Camera v2

6- Heatsink Case for Raspberry Pi4:

Used to keep the Raspberry Pi cool passively while the two fans cool the Pi actively for optimal cooling. [10] Figure 9 shows the Raspberry Pi 4 heatsink.

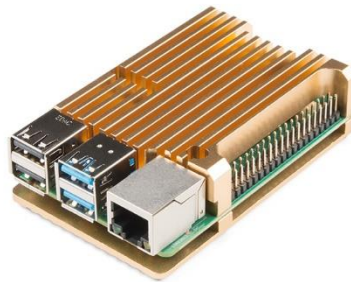


Figure 9 Heatsink Case for Raspberry Pi4

7- Thermal printer:

A thermal printer is a printer that uses heating head-on thermal papers to produce images. Its quality of print, ease of installation, minimum use of external components, and low power consumption, makes it more popular among token or ticket counters, grocery stores, entertainment

centers, healthcare, corporate industries, etc. It's both eco-friendly as well as a cost-effective solution for small-scale use as no ink Cartridge or ribbon is used. [11] Figure 9 shows the thermal printer.



Figure 10 Thermal printer

We use the thermal printer to print a QR code and then stick it on the product.

8- Impulse Sealer:

An impulse sealer, also called an impulsive sealer, is an electrically-powered tool that seals plastic packages. Plastic packaging is convenient for shipping and packing edible items, hardware, and other small components. An impulse sealer can make all the difference for packaging, protecting, and safely delivering your high-quality goodies. [12] Figure 11 shows the impulse sealer.



Figure 11 Impulse Sealer [16]

Depending on our needs in the project, we have to use an impulsive sealer to close the bags of fruit.

9- Linear Actuator:

An electric linear actuator is an electromechanical device mainly composed of a motor, a set of gears, and a motion mechanism in the form of a worm and tube. It converts the motor's rotary motion into linear motion by driving the gears and worm gear. It can be integrated into any equipment to push, pull, lift, lower, position or orient a load. [13] Figure 12 shows the Linear Actuator.



Figure 12 Linear Actuator [18]

Chapter 3

Design

3.1 Introduction

The purpose of the system design is to supplement the system architecture by providing information and data useful and necessary for implementation of the system elements. Design definition is the process of developing, expressing, documenting, and communicating the realization of the architecture of the system through a complete set of design characteristics described in a form suitable for implementation. [14]

3.2 Modeling Diagrams

In the following sections we used some modeling diagrams to clarify the system design.

3.2.1 Block Diagram

We represent the whole process of the system from the start to end, and specifying the input and the output of each component, and split the process into 3 phases depending on processes done in each phase as shown in figure 13.

3.2.2 Flow Charts

A detailed flowchart represented in figure 14 shows a that describes the main processes and main operations of the system, starting with the fruit image as an input moving through processing and calculations processes then the results printed over an LCD, then the system will do barcode printing and thermal packaging processes, finally the sealed fruit will be the last output of the system.

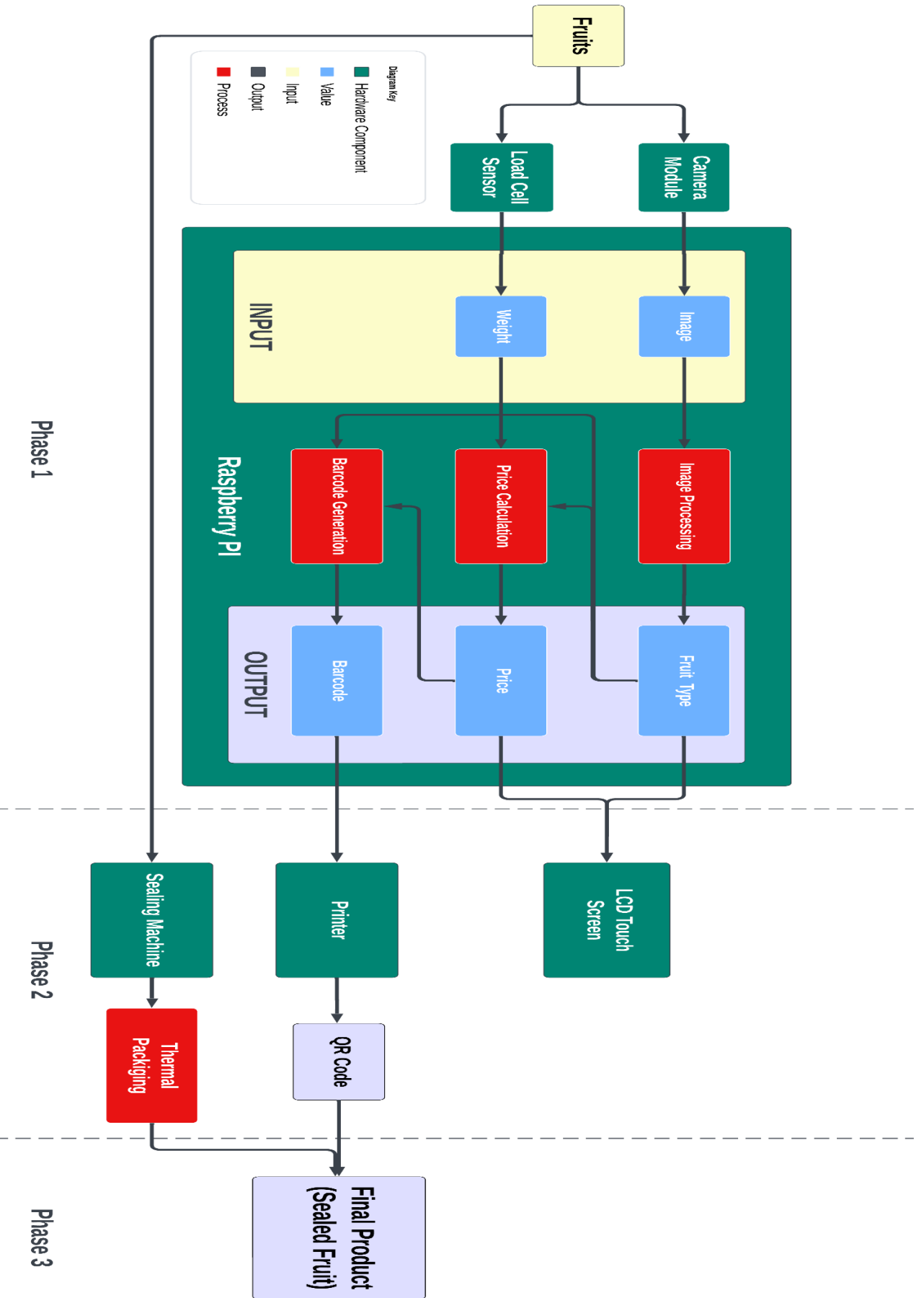


Figure 13 System Block Diagram



Figure 14 System Flow Chart

3.3 Software design

In the following sections we will describe the basic software operations and specify the software details of the system in term of figures and calculations.

3.3.1 Object Detection with Yolov8

In our System, we leverage the power of YOLOv8 (You Only Look Once, version 8) for robust and efficient object detection. YOLOv8 is a state-of-the-art real-time object detection algorithm that excels in accuracy and speed, making it an ideal choice for our image processing tasks. [15]

We determine to use a ready dataset from Kaggle named (Fruits and Vegetables Image Recognition Dataset), later on we will show how we deal with the dataset, and prune it to be ready for our model. [16]

3.3.1.1 Advantages of YOLOv8

Real-time Processing: YOLOv8's efficiency allows for real-time object detection, ensuring swift and accurate identification of fruits and vegetables as they are placed on the Smart Cashier System.

Accuracy: The model's training on a custom dataset enhances its accuracy in recognizing a diverse range of produce, contributing to the system's reliability.

Flexibility: YOLOv8 is known for its adaptability to various hardware configurations, making it a suitable choice for integration with the Raspberry Pi 4.

3.3.2 QR code generating

In our System, we have opted for QR codes as a means of efficient and versatile data representation. QR codes offer advantages such as high data capacity, fast scanning, and flexibility in encoding various types of information. This section outlines the implementation details of the QR code generation process within our system. [17]

3.3.2.1 Advantages of QR Codes

High Data Capacity: QR codes can store a significant amount of data, allowing for the inclusion of detailed product information within a small graphical space.

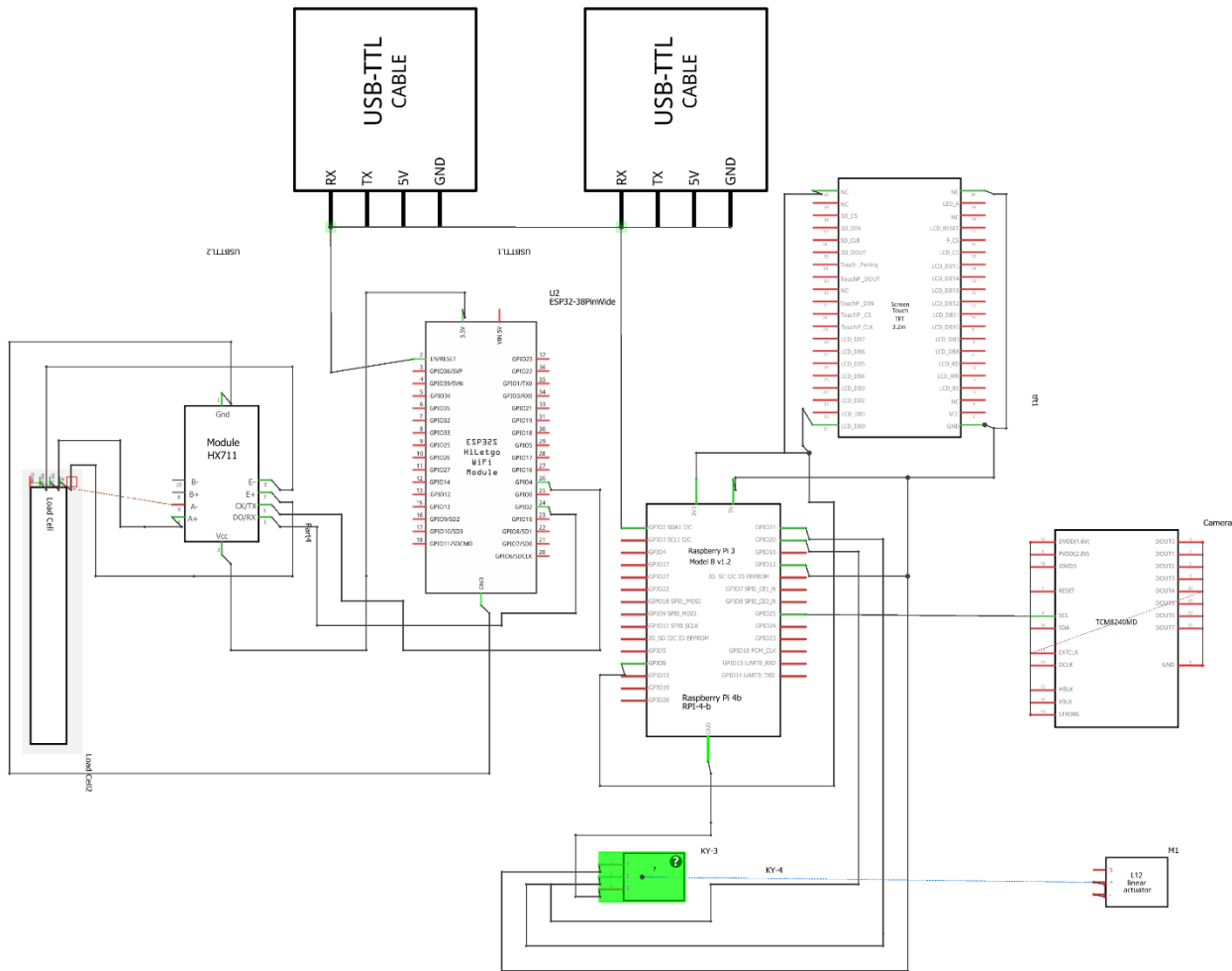
Quick Scanning: QR codes can be scanned quickly, streamlining the checkout process for both the cashier and the customer.

Versatility: QR codes can be utilized for various purposes, making them a versatile solution for our Smart Cashier System.

Dynamic Updates: In the future, additional features or information can be easily incorporated into the QR code without altering the physical representation on the product.

3.4 Schematic Design

The following figure shows the schematic design for the project, designed using Fritzing.



fritzing

Figure 15 Schematic Design

3.5 Exterior design

The following figure shows the exterior design for the project, designed using Sketchup.

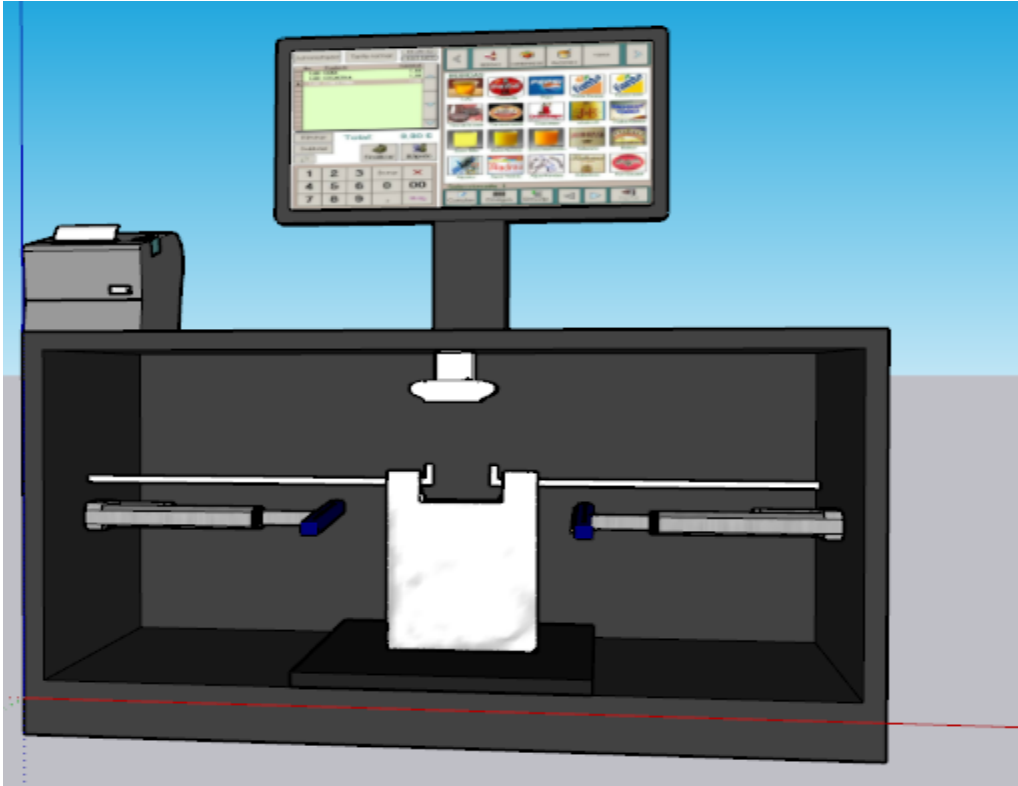


Figure 16 System Exterior Design

Chapter 4

Implementation and Results

4.1 Overview

This chapter describes the implementation part of our project with more details. It dives deep into the different hardware components of the system and its software with all of its modules.

4.2 Hardware Implementation

The main computer in Smart Cashier System will be the Raspberry PI4, which will get the name of the detected fruit/vegetable through image processing, and the weight of it from the loadcell using a microcontroller (Esp32), the raspberry PI4 and Esp32 are connected through a USB/Serial connection (The Esp32 code print a value on serial monitor then the Raspberry Pi captures the value through the serial connection) , then display them with the price on the User Interface (UI) using touchscreen, and after the user finished using the UI, the Raspberry PI4 will give the linear actuator a command using relay, to begin the thermal packaging process. Figure 16 describe how components connect to each other.

In the following subsections we will discuss the main circuits of the system separately, and configuring the wiring of each circuit. Figure 17 and figure 18 show the raspberry pi pinout and Esp32 pinout, which we used as a reference for connecting subsequent circuits.

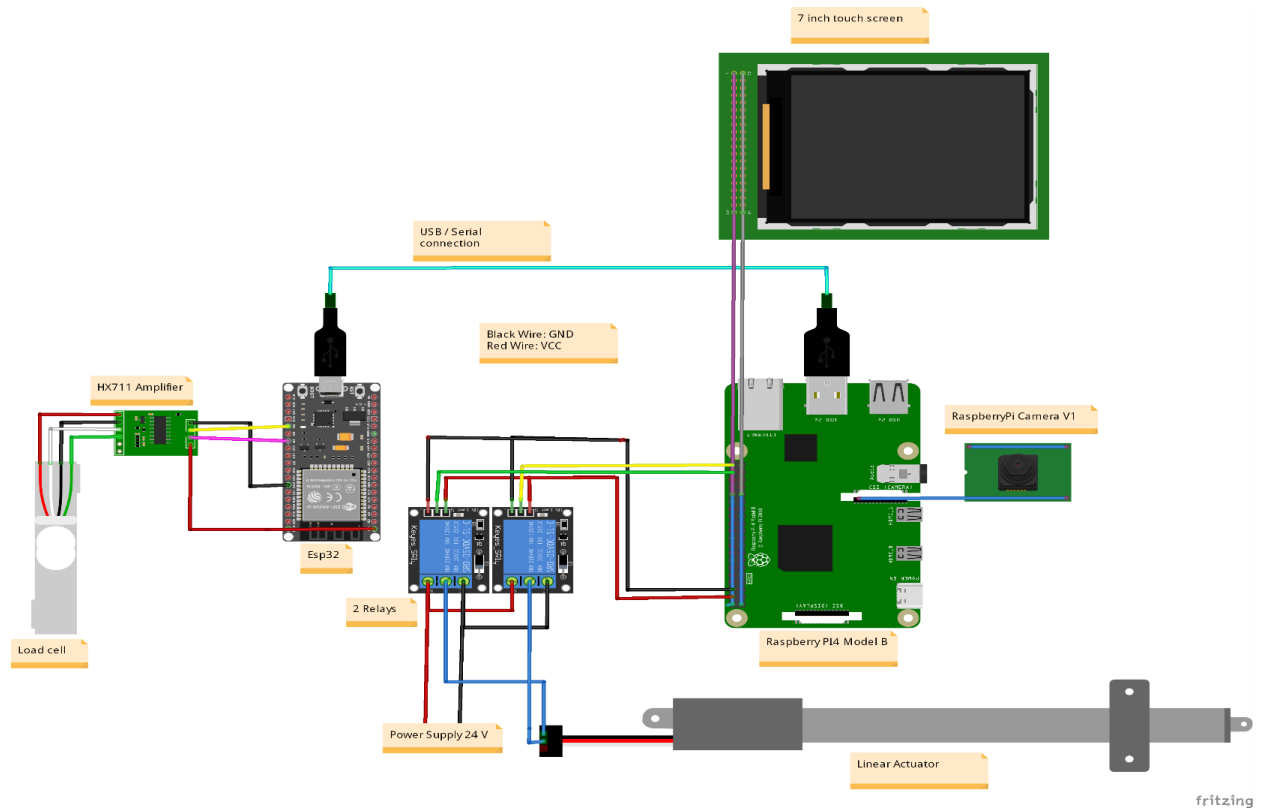


Figure 17 System Components

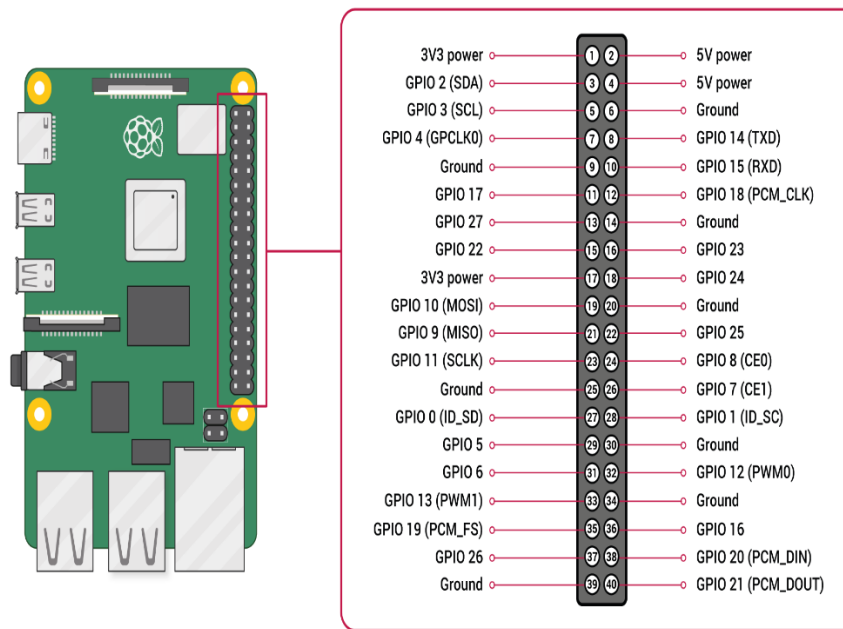


Figure 18 Raspberry Pi4 Pinout [23]

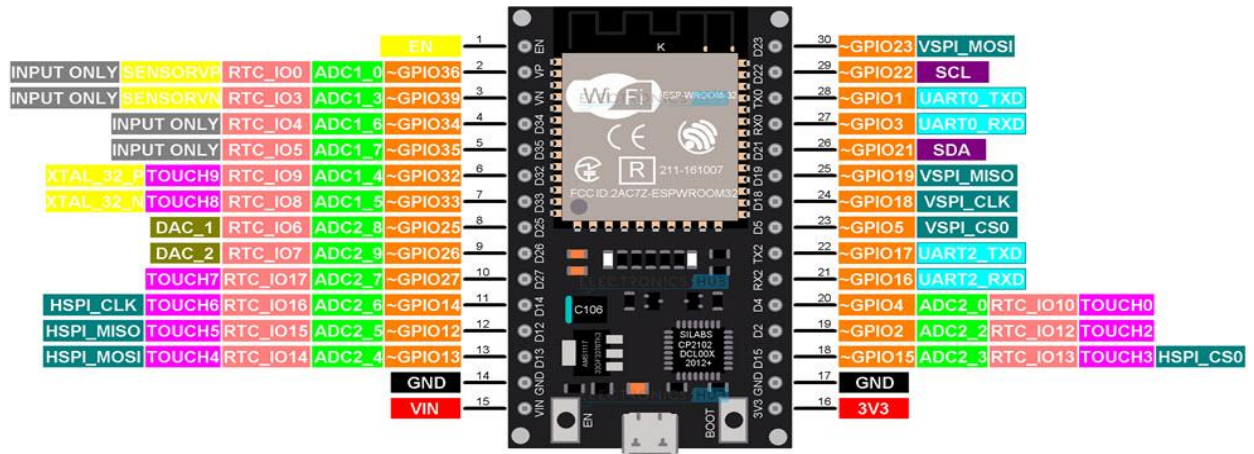


Figure 19 Esp32 Pinout [24]

4.2.1 Raspberry PI Circuit

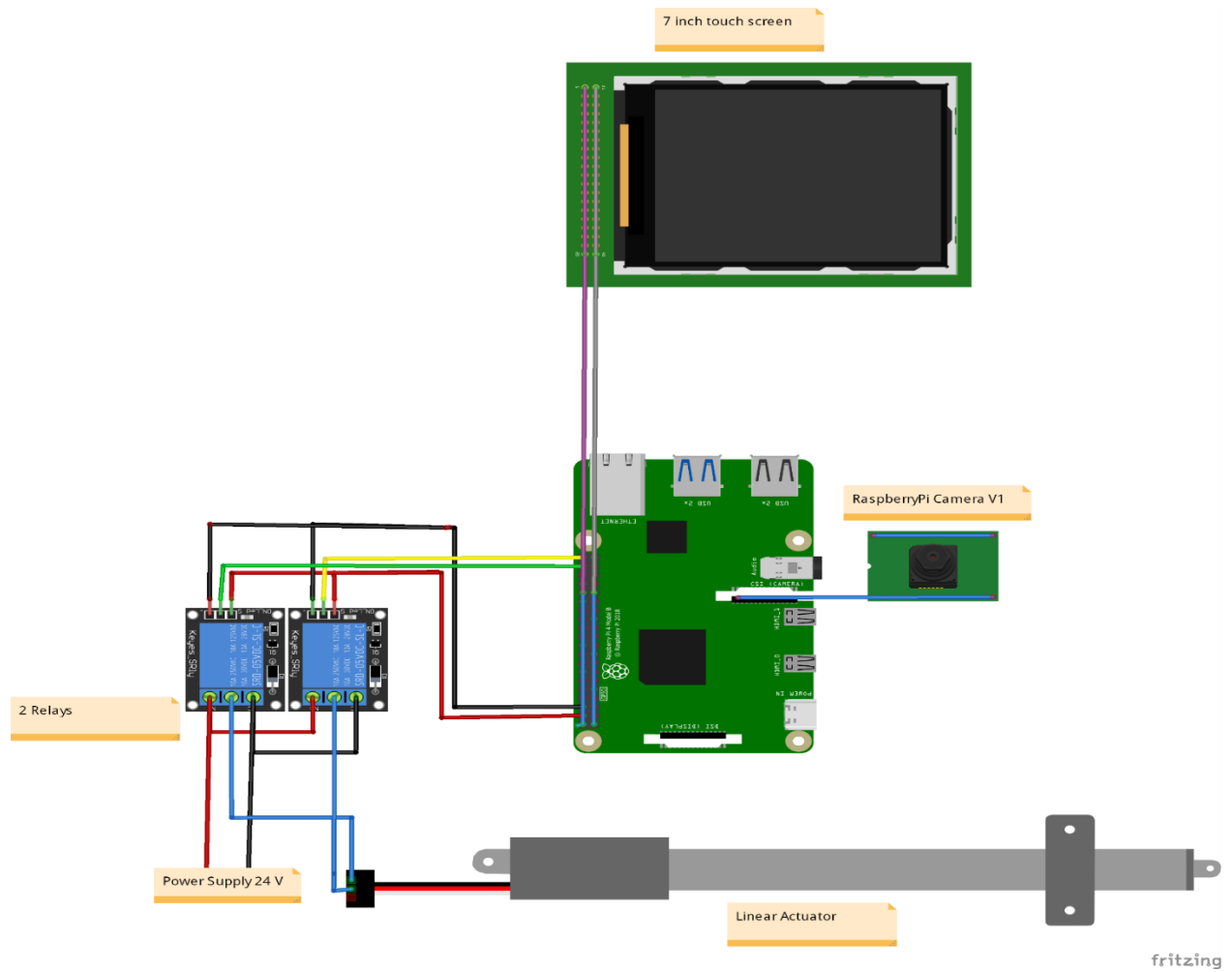


Figure 20 Raspberry PI4 Circuit

Figure 19 showing how the component are connected to the raspberry pi. First, we connect the camera to the raspberry pi through the camera flat cable, next we connect the active-low relays to the linear actuator, and connect the relays to the raspberry pi GPIO's, finally we connect the LCD touchscreen to the right pins on raspberry PI. Table 3 shows the wiring of each component due to raspberry PI GPIO's, and the wiring between components due to each other.

Table 3 Raspberry PI4 Component Wiring

Relay Pins	Raspberry PI Pins	Relay Pins	Linear Actuator Pins	Raspberry PI Pins (Physical pins)	LCD touchscreen Pins
E1	GPIO 21 (Yellow Wire)	Com 1	Pin 1	From Pin 1 to Pin 26	26 Pins
E2	GPIO 22 (Green Wire)	Com 2	Pin 2		
GND	GND (Black Wire)	NO1&NO2	+24 V power supply		
VCC 5V	5V (Red Wire)	NC1&NC2	-24 V power supply		

Note: Every two adjacent columns of the same color represent the way to connect the shown component

4.2.2 Esp32 Circuit

Figure 20 showing how the component are connected to the Esp32. First, we connect the load cell to the HX711 Amplifier then we connect the Amplifier to the raspberry PI, table 4 showing the wiring for each component due to Esp32 Pins.

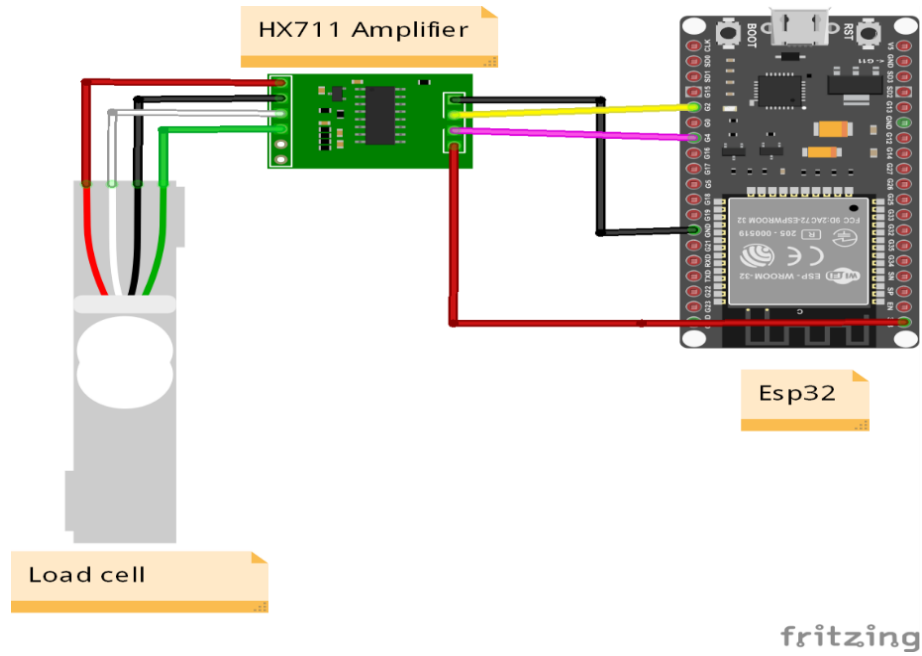


Figure 21 Esp32 Circuit

Table 4 Esp32 Component Wiring

Load cell	HX711	HX711	ESP32
E+ (Red wire)	E+	GND	GND (Black Wire)
E- (Black wire)	E-	DT	GPIO 5 (Yellow Wire)
A- (White wire)	A+	SCK	GPIO 4 (Pink Wire)
A+ (Green wire)	A-	VCC	3.3 V (Red Wire)

4.3 Software Implementation

This section describes the implementation details of software components of the system. It also explains the features and functions of each software component in the system.

4.3.1 Raspberry Pi Setup

in the following subsections we will discuss the first steps of the software implementation which are basic configuration of the raspberry pi.

4.3.1.1 Operating System

Due to dependency issues we choose to use Bullseye 64bit version of raspberry Pi OS, we download it through Raspberry Pi Imager, and install it to the raspberry pi using SD card.

4.3.1.2 VNC Viewer

In order to give us the ability to use the Raspberry Pi without connecting it to a screen, we decided to prepare a program that broadcasts the Raspberry Pi screen remotely, to facilitate working on the Raspberry Pi using a laptop.

The program name is VNC viewer, first we have to enable VNC through raspberry Pi interfaces.

Figure 21 showing how to enable the VNC interface in the raspberry pi.

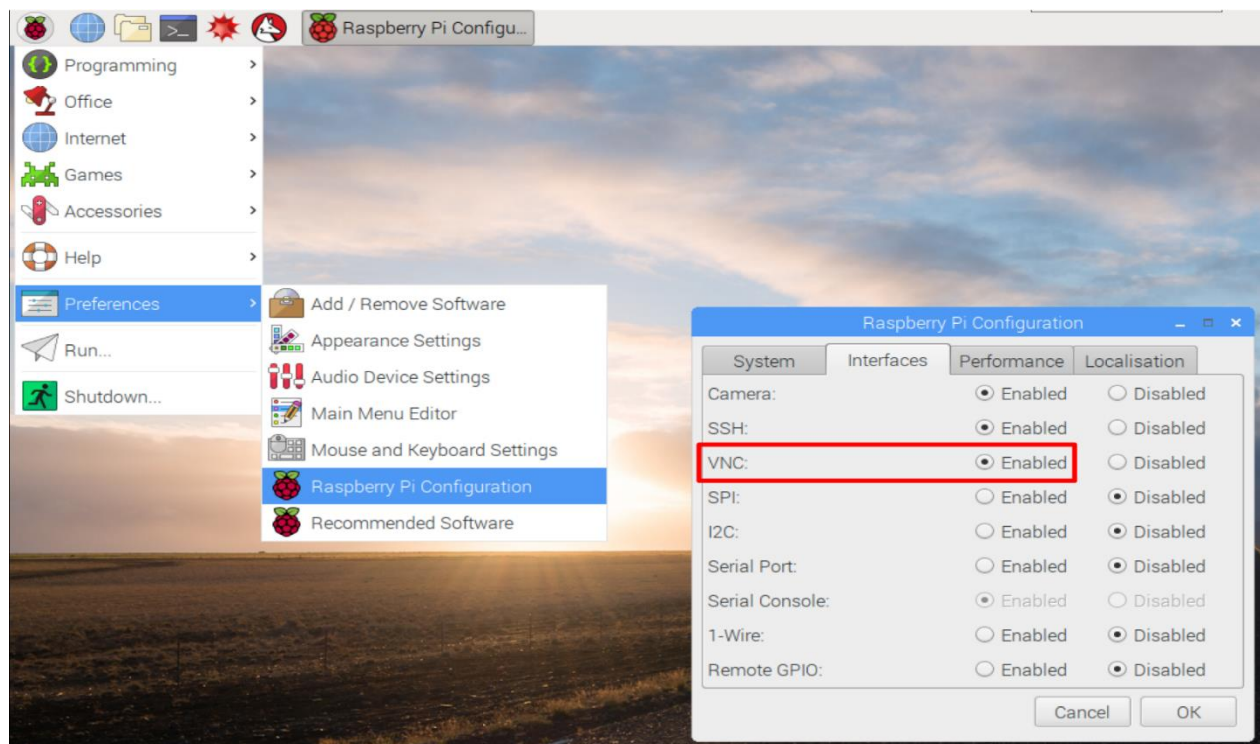


Figure 22 VNC configuration

4.3.1.3 Raspberry pi Camera setup

Due to compatibility issues with the Python Picamera module which is currently not compatible with the latest version of Raspberry Pi OS (Bullseye). [18]

To use the Picamera module, we will need to enable legacy support for the camera using these steps:

- 1- Open a terminal window and type the following command:



Figure 23 Raspberry Pi Settings

- 2- Use the cursor keys to scroll down to Interface Options and press the 'Enter' key.

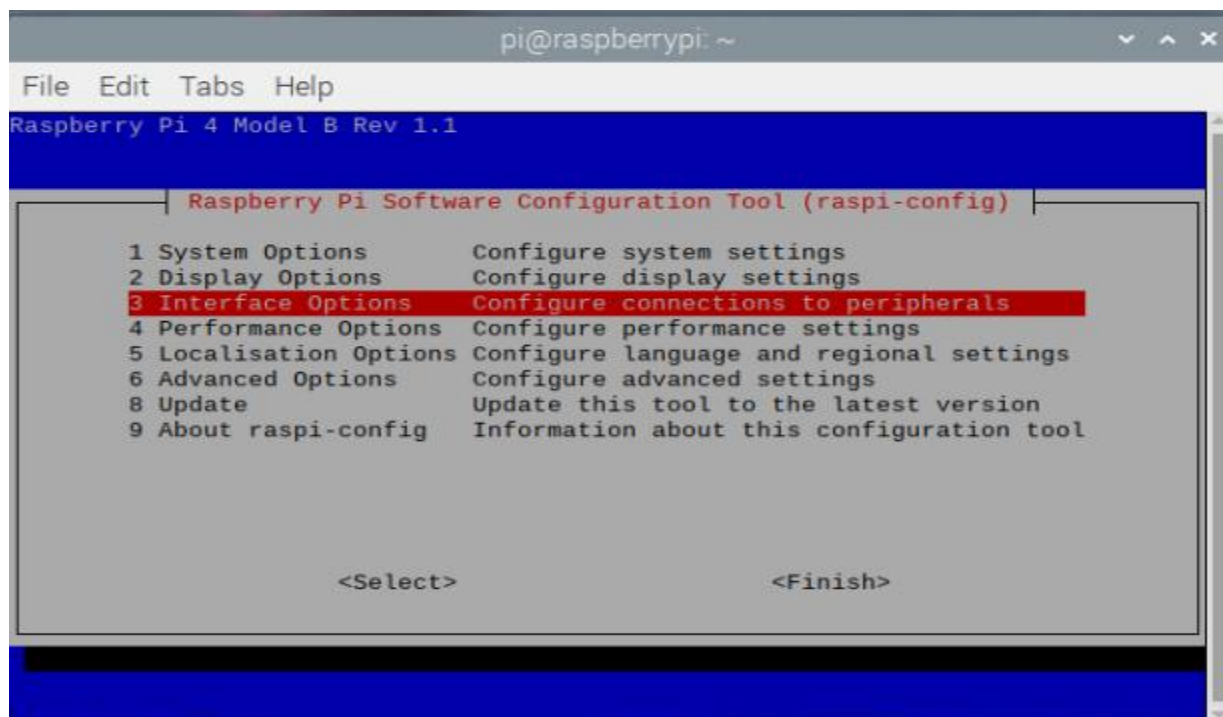
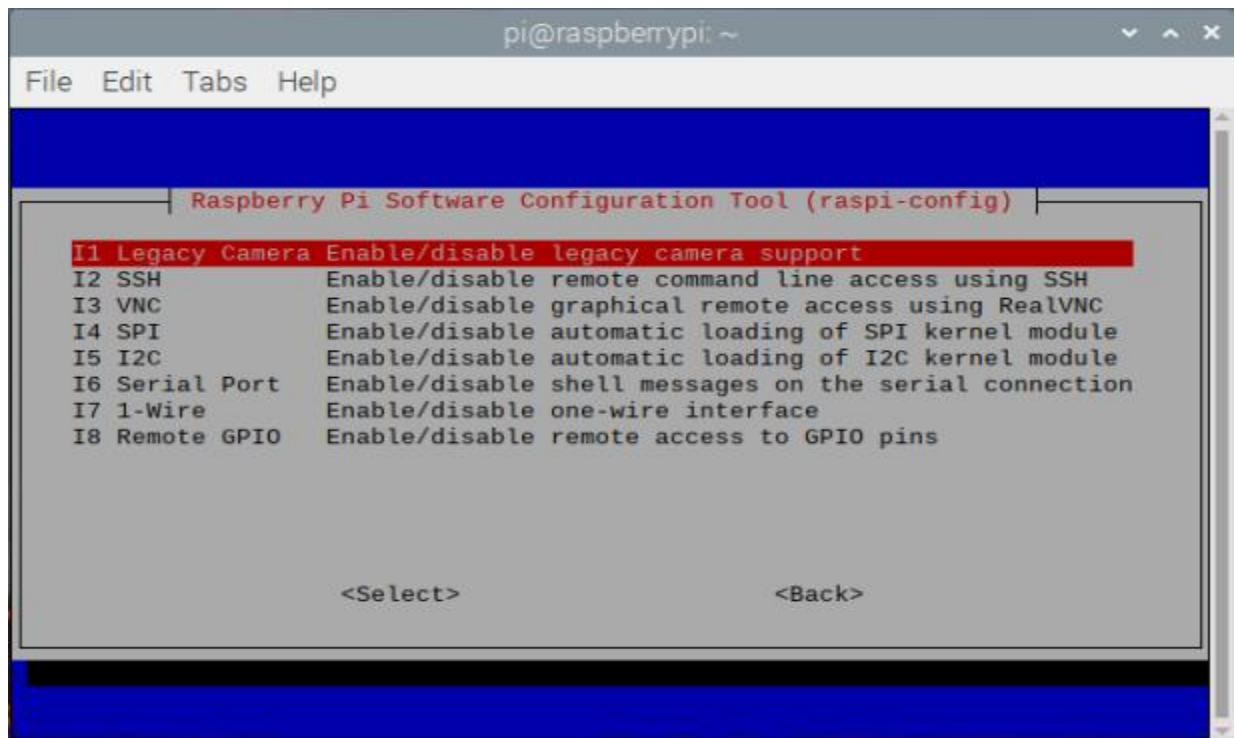


Figure 24 Enabling Raspberry Pi camera steps

- 3- Make sure 'Legacy Camera Enable/disable legacy camera support' is selected and press the 'Enter' key.



- 4- Use the cursor keys to select **<Yes>** and press the 'Enter' key
- 5- Press 'Enter' again to confirm.

After enabling the camera, we use this code which contain a terminal command to capture an image instead of using a library due to compatibility issues with the OS version.

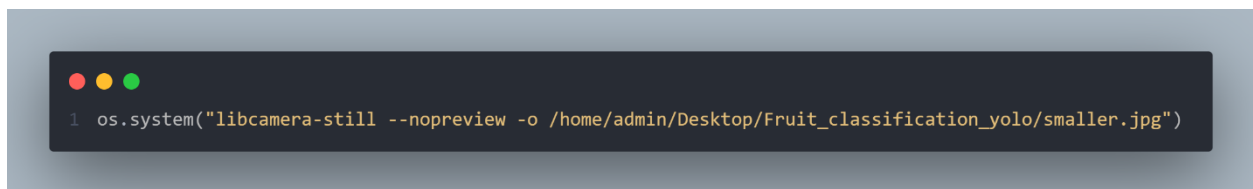


Figure 25 Capture an image command

4.3.1.4 Touchscreen Calibration

In order to use the touch feature in the screen, we must calibrate it. We carried out the calibration process by following the official manual of the screen manufacturer. [19]

Touch calibration

The display can be calibrated via xinput-calibrator.

1. Execute the following command to install the relevant software:

```
sudo apt-get install xserver-xorg-input-evdev xinput-calibrator
```

If the execution fails, you can check here [#Some possible problems](#)

2. Execute the following commands:

```
sudo cp -rf /usr/share/X11/xorg.conf.d/10-evdev.conf /usr/share/X11/xorg.conf.d/45-evdev.conf
sudo nano /usr/share/X11/xorg.conf.d/99-calibration.conf
```

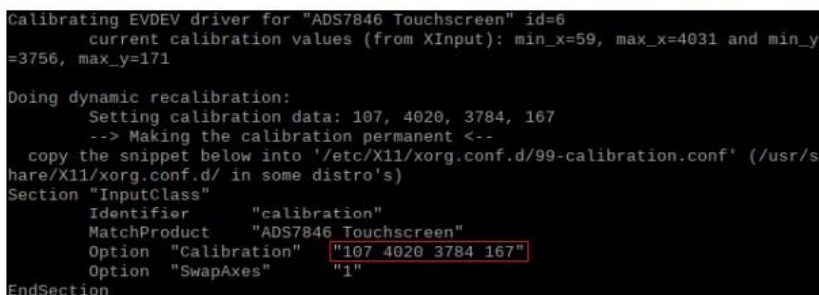
Add the following code to 99-calibration.conf:

```
Section "InputClass"
    Identifier      "calibration"
    MatchProduct   "ADS7846 Touchscreen"
    Option "Calibration" "73 4007 3976 84"
    Option "SwapAxes" "1"
    Option "EmulateThirdButton" "1"
    Option "EmulateThirdButtonTimeout" "1000"
    Option "EmulateThirdButtonMoveThreshold" "300"
EndSection
```

3. After reboot, touch will work normally under normal circumstances. But for different resistance screens, the accuracy of using the default calibration parameters may not be very suitable.

You can perform touch calibration by clicking the Raspberry Pi icon on the taskbar, selecting Preferences -> Calibrate Touchscreen, and following the displayed prompts.

4. After calibration, the following data will be displayed. If you want to save these touch values, you can replace the data in the red circle with the data in the corresponding position in 99-calibration.conf.



```
Calibrating EVDEV driver for "ADS7846 Touchscreen" id=6
current calibration values (from XInput): min_x=59, max_x=4031 and min_y
=3756, max_y=171

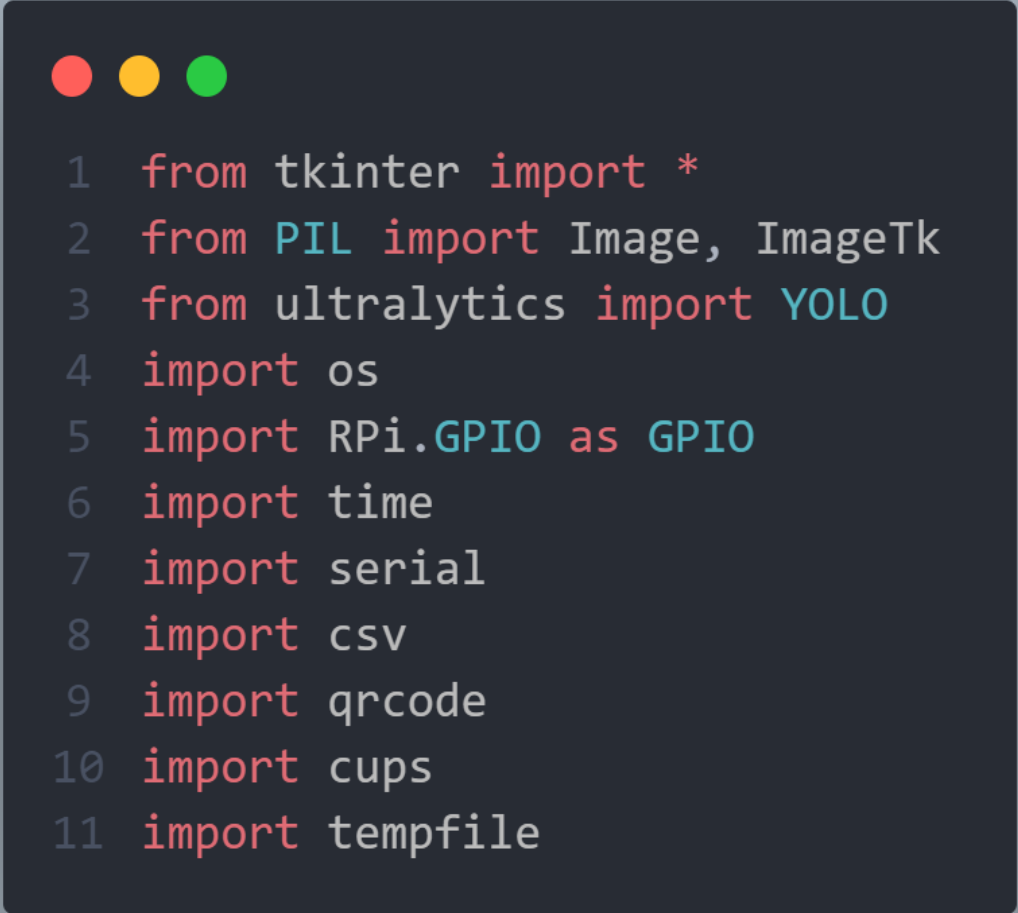
Doing dynamic recalibration:
Setting calibration data: 107, 4020, 3784, 167
--> Making the calibration permanent <--
copy the snippet below into '/etc/X11/xorg.conf.d/99-calibration.conf' (/usr/s
hare/X11/xorg.conf.d/ in some distro's)
Section "InputClass"
    Identifier      "calibration"
    MatchProduct   "ADS7846 Touchscreen"
    Option "Calibration" "107 4020 3784 167"
    Option "SwapAxes" "1"
EndSection
```

Figure 26 Touchscreen Calibration [26]

4.3.1.5 Installing Libraries

In order to make everything work well with our code, we used these libraries, each of them has its own function as shown in the Figure 24.

- 1- tkinter: Used for creating the graphical user interface (GUI).
- 2- PIL: Utilized for image processing and displaying images within the GUI.
- 3- ultralytics: Used for object detection with the YOLO (You Only Look Once) model.
- 4- os: Employed for executing system commands to capture images and interact with the filesystem.
- 5- RPi.GPIO: Facilitates interaction with the GPIO pins on the Raspberry Pi for controlling hardware.
- 6- time: Utilized for introducing delays during certain actions, such as controlling hardware.
- 7- serial: Used for serial communication, specifically for reading data from a connected device.
- 8- csv: Utilized for reading price data from a CSV file.
- 9- qrcode: Employed for generating QR codes representing product information.
- 10- cups: Used for interfacing with the Common Unix Printing System (CUPS) for printing QR codes.
- 11- tempfile: Used for creating temporary files, particularly for generating temporary PDF files when printing QR codes.

A terminal window with a dark background and three colored window control buttons (red, yellow, green) at the top left. The terminal displays 11 lines of Python code for importing various libraries.

```
1 from tkinter import *
2 from PIL import Image, ImageTk
3 from ultralytics import YOLO
4 import os
5 import RPi.GPIO as GPIO
6 import time
7 import serial
8 import csv
9 import qrcode
10 import cups
11 import tempfile
```

Figure 27 Used Libraries

4.3.2 Image Processing

We will discuss the image processing process in the next subsections.

4.3.2.1 Dataset Preparation

As we mentioned previously, we used a ready-made data set from Kaggle. After that, we performed a pruning process for this data set, and kept the 5 categories that would be tested on it which are (apple, banana, cucumber onion, potato, tomato). The pruning process was done through the online Roboflow tool.

Figure 21 showing the dataset through the pruning process.

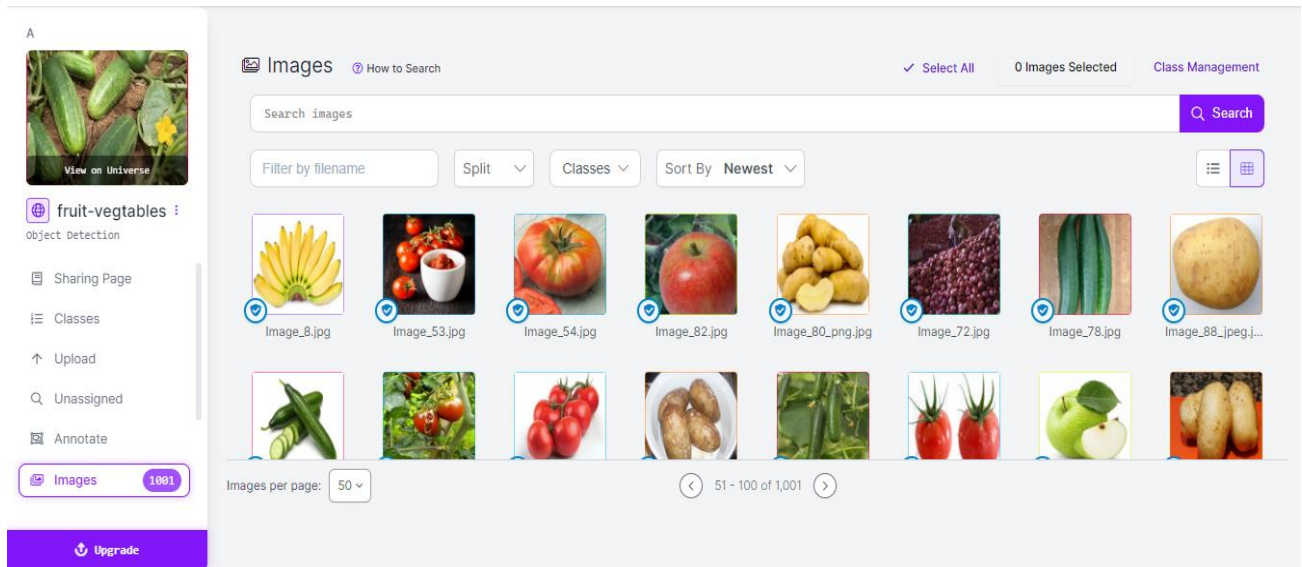


Figure 28 Data set Pruning

After that we export the dataset in TXT annotations and YAML config to use it with YOLOv8

4.3.2.2 Train the Model

Since our project will use Yolov8 algorithm for image processing, yolo comes with several pretrained models on coco dataset, each model has it is specifications, the following table represents the available models.

Table 5 Yolov8 Models [20]

Model	size (pixels)	mAP ^{val} 50-95	Speed CPU ONNX (ms)	Speed A100 TensorRT (ms)	parameters (M)
<u>YOLOv8n</u>	640	37.3	80.4	0.99	3.2
<u>YOLOv8s</u>	640	44.9	128.4	1.20	11.2
<u>YOLOv8m</u>	640	50.2	234.7	1.83	25.9
<u>YOLOv8l</u>	640	52.9	375.2	2.39	43.7
<u>YOLOv8x</u>	640	53.9	479.1	3.53	68.2

We choose Yolov8m model because it fulfilled our requirements, and it has moderate speed.

Since training the model requires a lot of GPU memory, we decided to train it using Google Colab because it provides a 12GB GPU memory and facilitates the process of training and installing libraries. [21]

The model was trained using 100 epochs, the following figure shows the training process using google colab notebook.

Epoch	GPU_mem	box_loss	cls_loss	dfl_loss	Instances	Size
1/100	6.42G	0.453	1.973	1.172	19	640: 100% 55/55 [00:31<00:00, 1.72it/s]
	Class	Images	Instances	Box(P	R	mAP50 mAP50-95): 100% 3/3 [00:03<00:00, 1.08s/it]
	all	87	87	0.642	0.707	0.707 0.613
Epoch	GPU_mem	box_loss	cls_loss	dfl_loss	Instances	Size
2/100	6.6G	0.3296	1.166	1.046	20	640: 100% 55/55 [00:27<00:00, 1.99it/s]
	Class	Images	Instances	Box(P	R	mAP50 mAP50-95): 100% 3/3 [00:01<00:00, 2.43it/s]
	all	87	87	0.671	0.366	0.416 0.291
Epoch	GPU_mem	box_loss	cls_loss	dfl_loss	Instances	Size
3/100	6.6G	0.3609	1.112	1.062	16	640: 100% 55/55 [00:26<00:00, 2.05it/s]
	Class	Images	Instances	Box(P	R	mAP50 mAP50-95): 100% 3/3 [00:01<00:00, 2.19it/s]
	all	87	87	0.539	0.715	0.66 0.574
Epoch	GPU_mem	box_loss	cls_loss	dfl_loss	Instances	Size
4/100	6.6G	0.3333	1.064	1.032	18	640: 100% 55/55 [00:28<00:00, 1.91it/s]
	Class	Images	Instances	Box(P	R	mAP50 mAP50-95): 100% 3/3 [00:01<00:00, 2.64it/s]
	all	87	87	0.236	0.367	0.276 0.148
Epoch	GPU_mem	box_loss	cls_loss	dfl_loss	Instances	Size
5/100	6.59G	0.3103	1.042	1.015	19	640: 100% 55/55 [00:27<00:00, 2.00it/s]
	Class	Images	Instances	Box(P	R	mAP50 mAP50-95): 100% 3/3 [00:01<00:00, 2.58it/s]
	all	87	87	0.438	0.543	0.603 0.342

Figure 29 Google Colab notebook

4.3.2.3 Deploying the model

After the training process finished the model is now ready to be exported in a (.pt) format and deployed on the raspberry pi, and in the next chapter we will show the evaluation and testing process of the model.

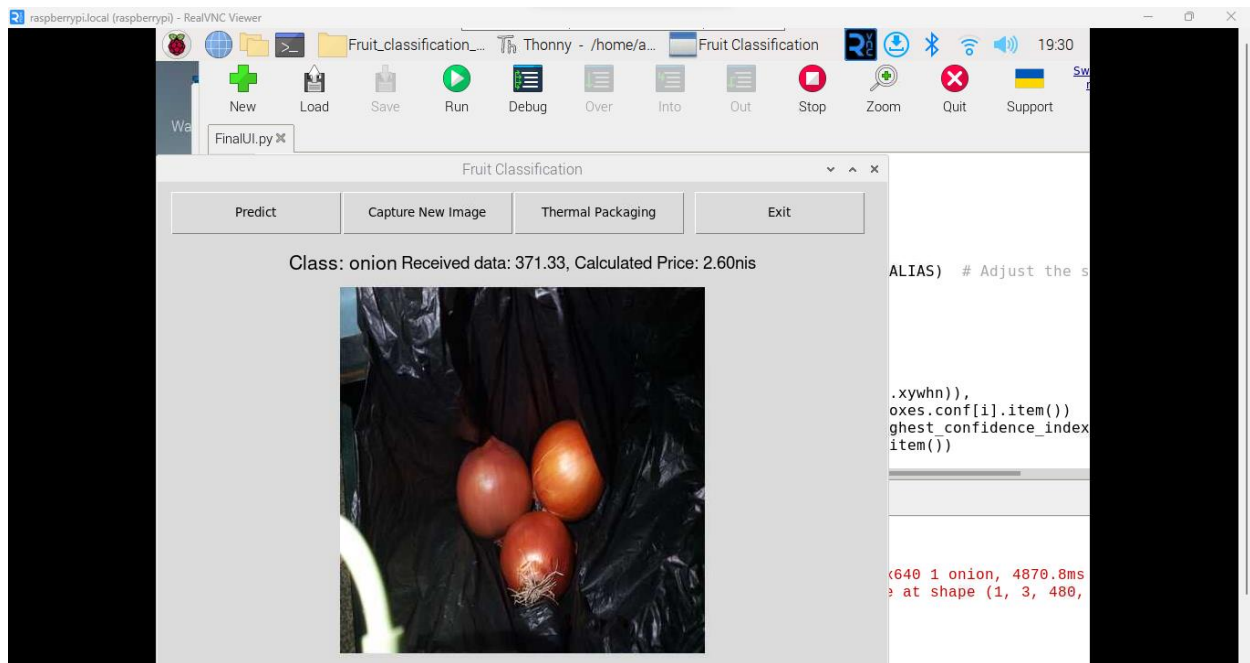
4.3.3 Esp32 Setup

The esp32 calculates the weight using a code, then prints the weight using Serial.println(Weight Value), so that the Raspberry Pi receives the value via USB/Serial connection.

4.3.4 System User Interface

After all the parts of software implementation is finished, the time to build the user interface is come, we used a pre-installed python library to install the UI, the library name tkinter, the UI contains several components including buttons and labels.

The following figure shows the UI with its components.



4.3.5 Coding

In the next sub sections, we will describe the important functions and lines of our code and explain them.

4.3.5.1 Load And Process The Prices File

This section of code loads pricing information from a CSV file named 'prices.csv'. It initializes an empty dictionary called prices. It then opens the CSV file in read mode using open() and creates a CSV reader object using csv.DictReader(). For each row in the CSV file, it extracts the class name and price information using the keys 'class_name' and 'price' respectively. It converts the price to a float type and populates the prices dictionary with class names as keys and their corresponding prices as values. Finally, it closes the CSV file. The code is shown in the following figure.

```

1 # load the prices .csv file
2 prices = {}
3 csvfile = open('prices.csv', newline='')
4 reader = csv.DictReader(csvfile)
5 for row in reader:
6     class_name = row['class_name']
7     price = float(row['price'])
8     prices[class_name] = price

```

Figure 30 Load And Process The Prices File

4.3.5.2 Load The YOLOv8 Model

The first part of the code loads a pre-trained YOLO model for object detection. It initializes an instance of the YOLO class with the path to the model file 'best.pt', typically stored at "best.pt". As shown in figure 29.

The second part defines a method update_class_label() used to update the predicted class label based on the captured image. It takes the captured image path as input, passes it through the YOLO model to make predictions, and extracts the class name with the highest confidence score from the predictions. It returns the predicted class name. As shown in figure 30

```

1 self.model = YOLO("/home/admin/Desktop/Fruit_classification_yolo/best.pt")

```

Figure 31 Load YOLO Model

```

1 def update_class_label(self):
2     predictions = self.model(self.captured_image_path)
3     highest_confidence_index = max(range(len(predictions[0].boxes.xywhn)),
4                                   key=lambda i: predictions[0].boxes.conf[i].item())
5     highest_confidence_prediction = predictions[0].boxes.xywhn[highest_confidence_index]
6     cls = int(predictions[0].boxes.cls[highest_confidence_index].item())
7     class_name = self.model.names[cls]
8     return class_name

```

Figure 32 Detect The Class of Fruit

4.3.5.3 Process the received data from Esp32

The method, `check_received_data()`, continuously reads data from a serial port which is the weight of the fruit in grams. It decodes the data, displays it in the UI, calculates the price based on the received data and class name, and updates the UI with the calculated price. It handles Unicode decoding errors and schedules itself to run every 10 milliseconds using `self.root.after()`. As shown in figure 31.

```

1 def check_received_data(self):
2     try:
3         data = self.ser.readline().decode('utf-8', errors='replace').strip()
4         if data:
5             print("Received data:", data)
6             self.received_data.set(f"Received data: {data}")
7             self.received_data_label.grid(row=0, column=1)
8
9             # Calculate the price and display it
10
11            price = self.prices[class_name]
12            calculated_price = (float(data) / 1000) * float(price)
13            self.calculated_price = calculated_price
14            self.received_data.set(
15                f"Received data: {data}, Calculated Price: {calculated_price:.2f}nis")
16        else:
17            print("No data received.")
18    except UnicodeDecodeError as e:
19        print(f"UnicodeDecodeError: {e}")
20
21    # Continue checking for received data every 10 milliseconds
22    self.root.after(10, self.check_received_data)
23

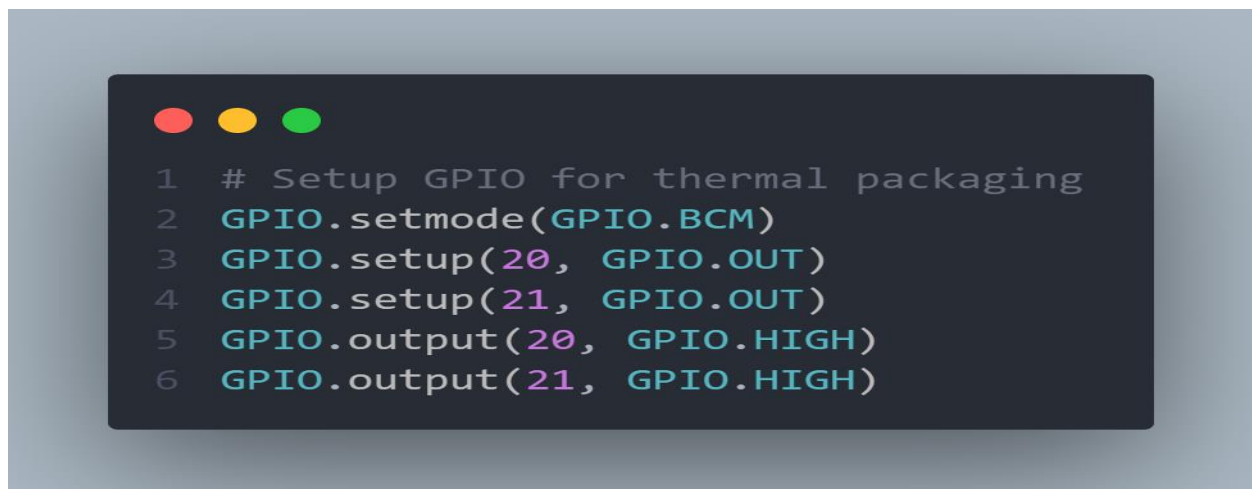
```

Figure 33 Read Weight Value

4.3.5.4 Thermal packaging processing

The first part initializes the GPIO pins for controlling thermal packaging. It sets the GPIO mode to Broadcom numbering, configures GPIO pins 20 and 21 as outputs, and sets their initial state to HIGH. As shown in figure 32.

The second part defines a method `thermal_packaging()` responsible for executing the thermal packaging process. It controls the GPIO pins to activate the thermal packaging mechanism in a specific sequence with defined time intervals. After the packaging is complete, it makes the "Finish and Print" button visible in the UI. As shown in figure 33

A terminal window with a dark background and three colored window control buttons (red, yellow, green) at the top left. The code is as follows:

```
1 # Setup GPIO for thermal packaging
2 GPIO.setmode(GPIO.BCM)
3 GPIO.setup(20, GPIO.OUT)
4 GPIO.setup(21, GPIO.OUT)
5 GPIO.output(20, GPIO.HIGH)
6 GPIO.output(21, GPIO.HIGH)
```

Figure 34 Setup GPIO Pins

A terminal window with a dark background and three colored window control buttons (red, yellow, green) at the top left. The code is as follows:

```
1 def thermal_packaging(self):
2     # Execute the thermal packaging code
3     GPIO.output(20, GPIO.LOW)
4     time.sleep(17.3)
5     GPIO.output(20, GPIO.HIGH)
6     time.sleep(1.5)
7     GPIO.output(21, GPIO.LOW)
8     time.sleep(20)
9     GPIO.output(21, GPIO.HIGH)
10    GPIO.output(20, GPIO.HIGH)
11    GPIO.output(21, GPIO.HIGH)
12
13    # Show the "Finish and Print" button after thermal packaging is done
14    self.finish_and_print_button.grid(row=0, column=4)
15
```

Figure 35 Thermal Packaging Code

4.3.5.5 Generate And Print QR Code

The first part of the code defines a method `finish_and_print(self)` responsible for generating a QR code containing class name, received data, and calculated price, displaying it in a new window, and providing an option to print it. It uses the `qrcode` library to generate the QR code, saves it as an image, creates a new window to display the QR code image, and adds a button to print the QR code. As shown in figure 34.

The second part defines a method `print_qr_code(self)` responsible for printing the generated QR code. It uses the `PIL` library to open the QR code image, converts it to PDF format, and prints it using the default printer configured in the CUPS system. Finally, it deletes the temporary PDF file after printing. As shown in figure 35.

```
1 def finish_and_print(self):
2     # Generate QR code with class name, received data, and calculated price
3     qr_data = f"{self.class_name.get()}, {self.received_data.get()}, {self.calculated_price:.2f}nis"
4     qr = qrcode.QRCode(
5         version=1,
6         error_correction=qrcode.constants.ERROR_CORRECT_L,
7         box_size=10,
8         border=4,
9     )
10    qr.add_data(qr_data)
11    qr.make(fit=True)
12
13    # Create a new window for displaying the QR code
14    self.qr_code_window = Toplevel(self.root)
15    self.qr_code_window.title("QR Code")
16
17    # Save the QR code image to a file
18    qr_code_image = qr.make_image(fill_color="black", back_color="white")
19    qr_code_image_path = "/home/admin/Desktop/Fruit_classification_yolo/qr_code.png"
20    qr_code_image.save(qr_code_image_path)
21
22    # Create a label to display the QR code image
23    self.qr_code_image_label = Label(self.qr_code_window)
24    self.qr_code_image_label.pack(pady=10)
25
26    # Load and display the QR code image
27    self.load_qr_code_image(qr_code_image_path)
28
29    # Create a button to print the QR code
30    print_button = Button(self.qr_code_window, text="Print",
31                          command=self.print_qr_code, height=2, width=20)
32    print_button.pack(pady=10)
```

Figure 36 Generate QR Code

```

1  def print_qr_code(self):
2      # Use PIL to open and print the QR code image
3
4      qr_code_image_path = "/home/admin/Desktop/Fruit_classification_yolo/qr_code.png"
5
6      # Temporary file for the PDF version of the QR code
7      temp_pdf = tempfile.NamedTemporaryFile(suffix=".pdf", delete=False)
8
9      try:
10         # Convert the QR code image to PDF using PIL
11         qr_code_image = Image.open(qr_code_image_path)
12         qr_code_image.save(temp_pdf.name, "PDF")
13
14         # Get the default printer name from CUPS
15         conn = cups.Connection()
16         default_printer = conn.getDefault()
17
18         # Print the PDF using CUPS
19         conn.printFile(default_printer, temp_pdf.name, "QR Code Print", {})
20         print("QR Code printed successfully.")
21
22     except Exception as e:
23         print(f"Error printing QR Code: {e}")
24
25     finally:
26         # Close and delete the temporary PDF file
27         temp_pdf.close()
28         os.unlink(temp_pdf.name)
29

```

Figure 37 Read The QR Code And Print It

Chapter 5

Testing and Validation

5.1 Overview

In this chapter we will discuss the testing of all components of the system and the results obtained. We tested all the parts to ensure that all of the functions work as expected and without errors.

5.2 Hardware Testing

In the following sub sections, we will present a test for each hardware component, and the test result of each component.

5.2.1 Weighing System Testing (Load cell testing)

In order to test our weighing, we choose different objects and measure their actual weight, then weighing them using our system and calculate the relative error, the following table show the whole process in terms of numbers.

Table 6 Weighing System Testing

Object	Actual Weight (kg)	Measured Weight (Using Our Weighing system) (kg)	Relative Error% = $\text{Measured} - \text{Actual} / \text{Actual}$ [22]
1kg of salt	1.012	1.011	0.098%
1kg of sugar	1.026	0.99055	3.455%
2kg of salt	2.024	2.027	0.148%
3kg of salt	3.030	2.980	1.650%
Average Relative Error			1.33%

5.2.2 Linear Actuator and Impulse Sealer

We examined both the actuator and sealer, and the test was conducted on the actuator to determine the time that takes to seal the packet, based in actuator speed that appears in the datasheet and several experiment, the forward time was 17.3 sec and the reverse time is 20 sec. For the impulse sealer, based on several experiments and tests, the temperature was set to level 3. The following figure shows the impulse process.



Figure 38 Linear Actuator and Impulse Testing

5.2.3 Touch Screen Testing

We worked on examining the screen by examining its touch screen after performing the calibration process, as we explained previously, and it turned out that the calibration process was completed properly, as the following image shows.

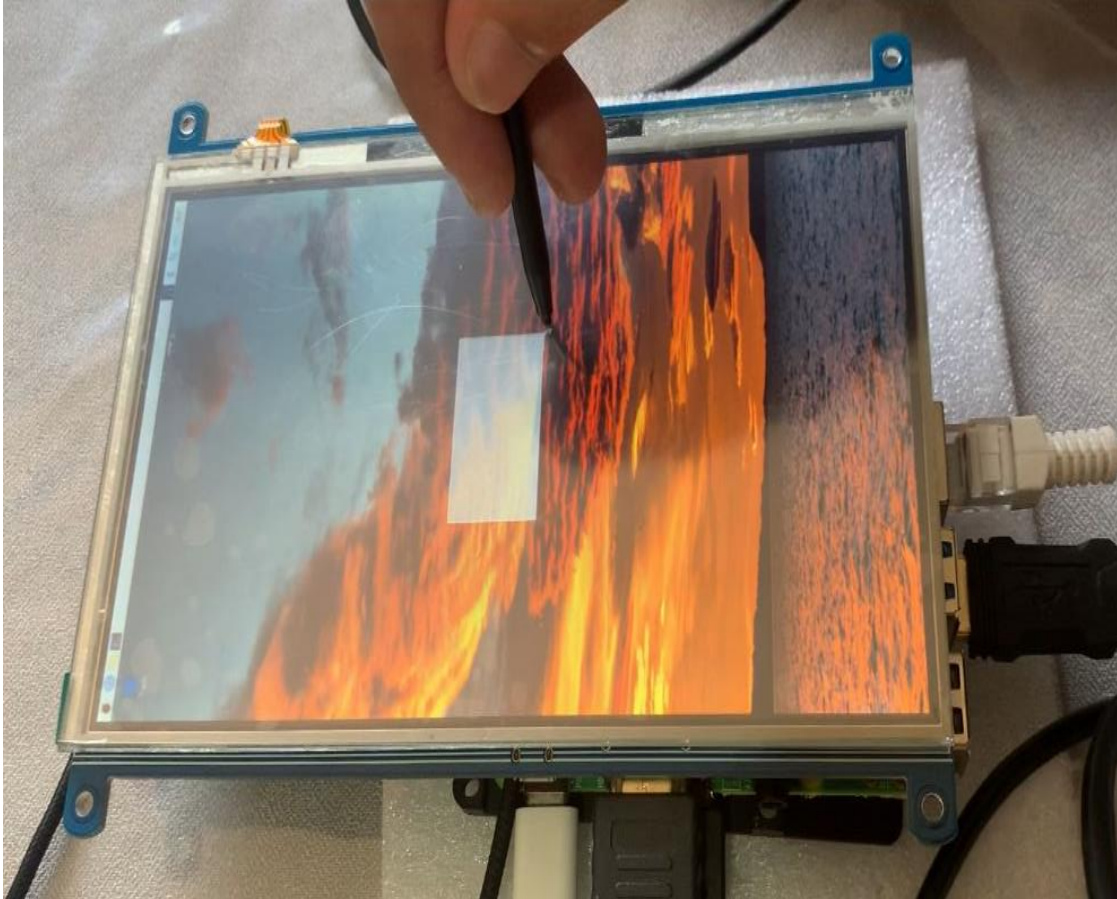


Figure 39 Touch Screen Testing

5.3 Software Testing

In the next sub sections, we will discuss the software testing, by perform a test for each software component, and test the functionality of each one of them.

5.3.1 Image Process Testing

In the following subsections we will go over the image processing and do the inspection for image processing components.

5.3.1.1 Evaluate The Model

we use different curves in order to evaluate our model, one of them was Precision-Recall Curve.

Precision is the proportion of correct positive classifications (true positive) divided by the total number of predicted positive classifications that were made (true positive + false positive). Recall is the proportion correct positive classifications (true positive) divided by the total number of the truly positive classifications (true positive + false negative). [23]

A PR curve is simply a graph with Precision values on the y-axis and Recall values on the x-axis. In other words, the PR curve contains $TP/(TP+FP)$ on the y-axis and $TP/(TP+FN)$ on the x-axis.

- It is important to note that Precision is also called the Positive Predictive Value (PPV).
- Recall is also called Sensitivity, Hit Rate or True Positive Rate (TPR).

The figure below shows the PR curve for our model.

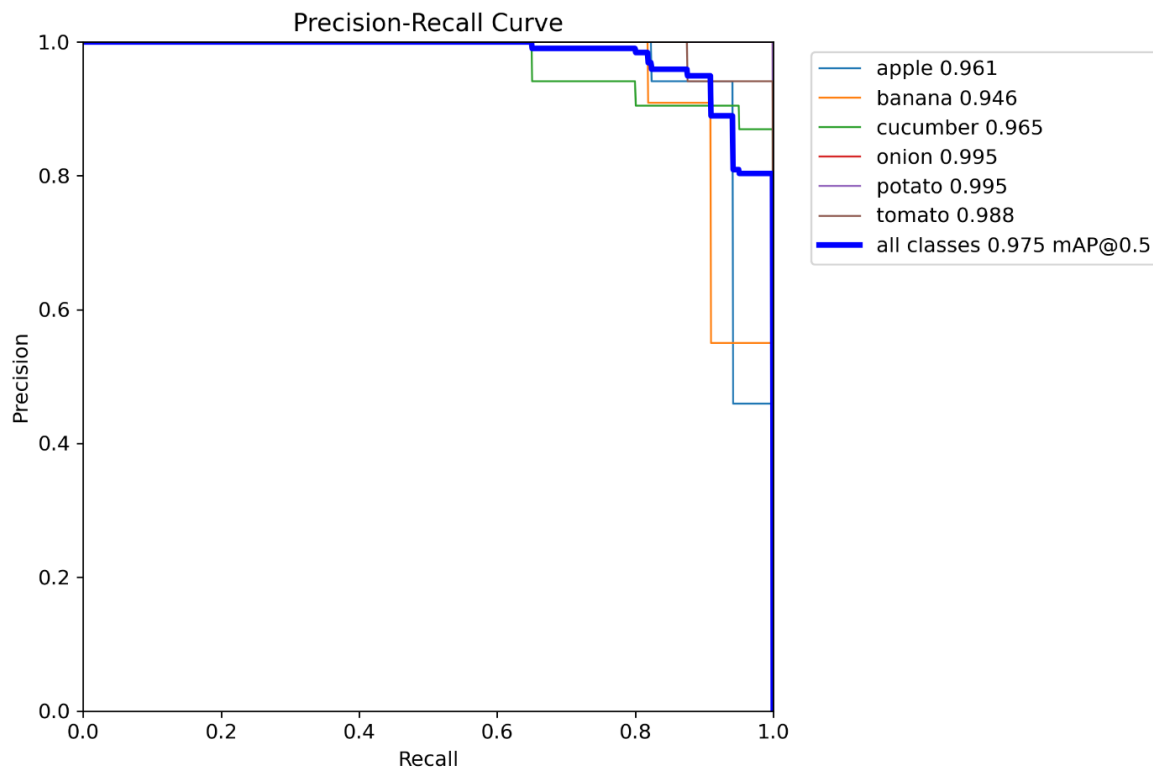


Figure 40 Precision-Recall (PR) Curve

The following curve represents the F1 score across various thresholds. Interpreting this curve can offer insights into the model's balance between false positives and false negatives over different thresholds. [24]

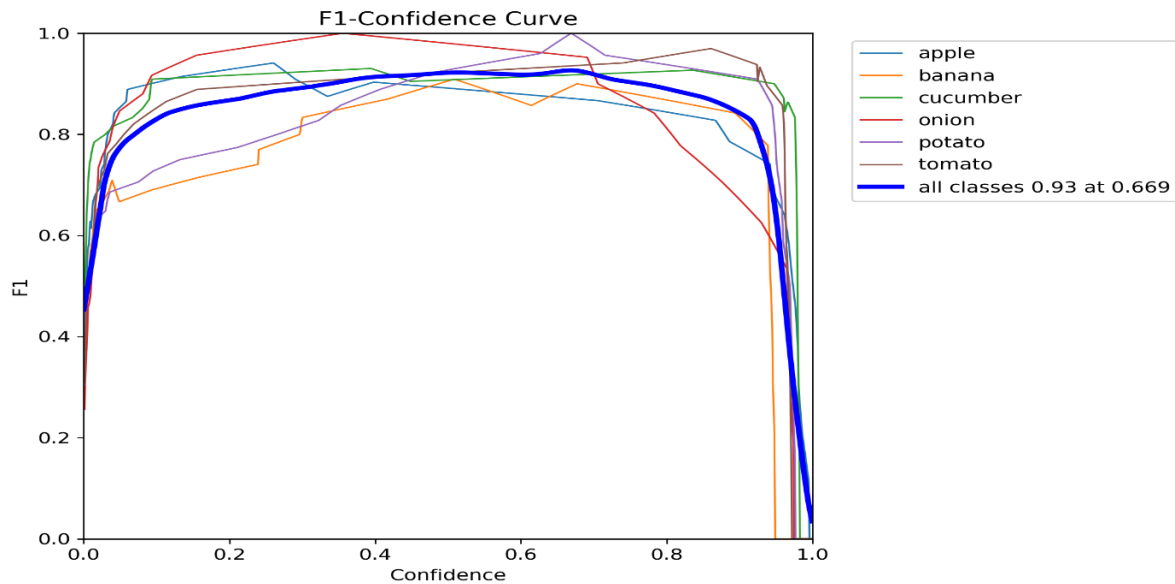


Figure 41 F1 Score Curve

Figure 41 present the confusion matrix, the confusion matrix provides a detailed view of the outcomes, showcasing the counts of true positives, true negatives, false positives, and false negatives for each class. [24]

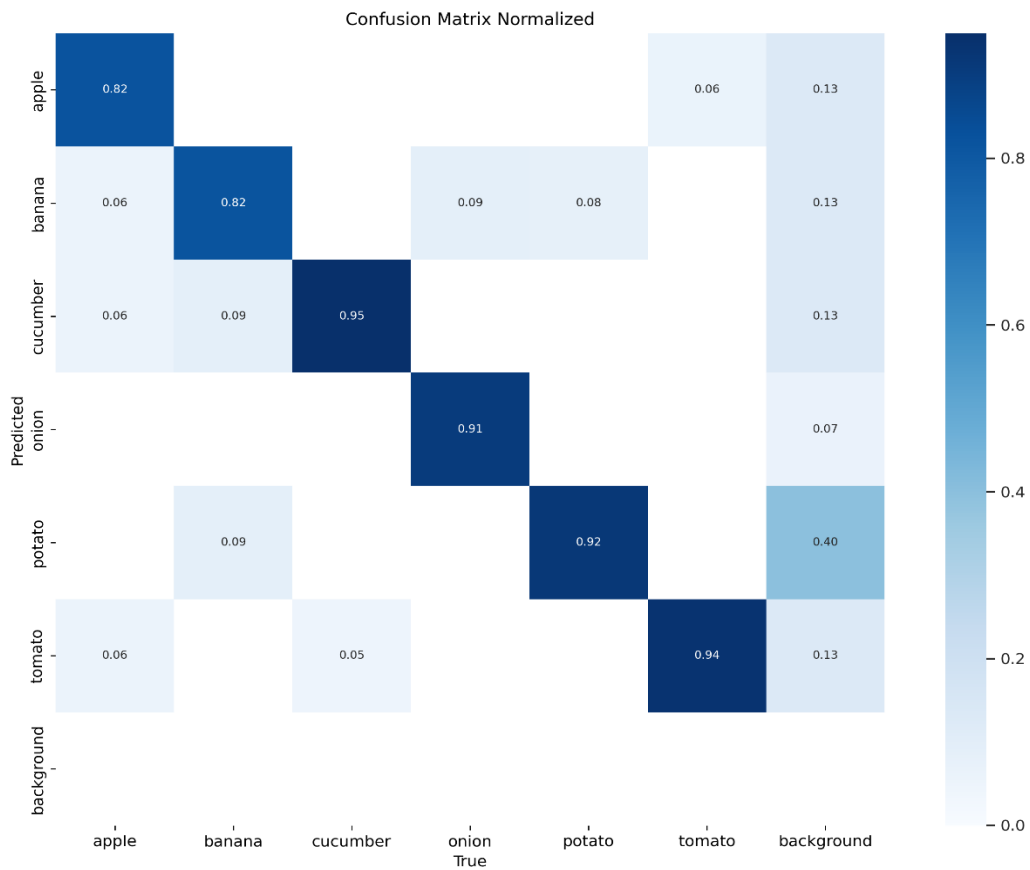


Figure 42 Confusion Matrix

5.3.1.2 Test The Model

We conducted an examination of the model by experimenting with some of the images on which it was trained, and the results appeared clear, as the confirmation rate for all images was very high more than 90%.



Figure 43 Model Validation

5.3.2 QR code Testing

We checked the QR code by scanning it after printing it, and the result of the scan appeared as shown in the following figure.

The price appears twice to make the verification process clearer and more reliable, and easy to get the information. The QR contains different values:

- Class: the prediction of the fruit name
- Received data: represent the weight in grams.
- Calculated Price: the final price for the fruit.

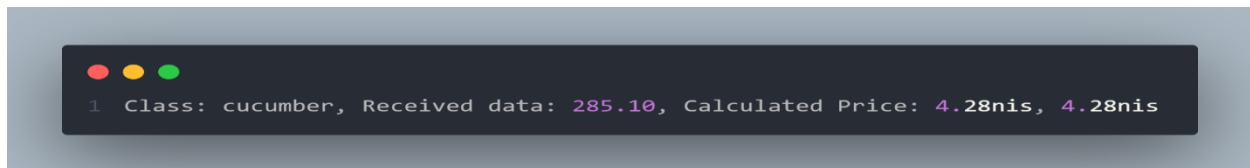


Figure 44 QR Code Content

5.3.3 UI Testing and Functional Testing

We conducted an inspection of the interface by testing the pressure on all the buttons and ensuring that they work efficiently and perform the required function.

The following figure show the UI components.

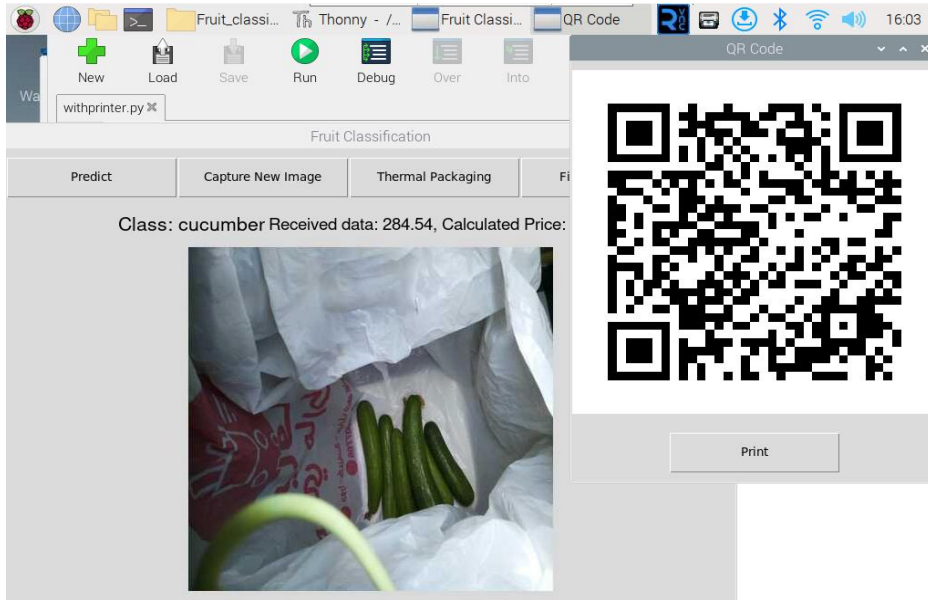


Figure 45 Functional Testing

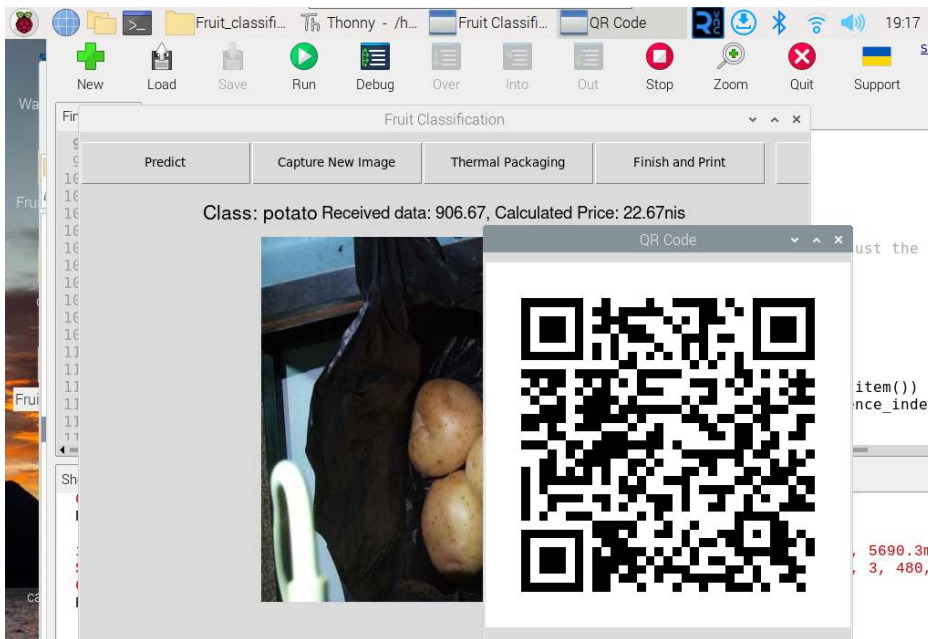


Figure 46 UI components Testing

Chapter 6

Conclusion and Future Work

6.1 Summary of Deliverables

The project successfully delivered a functional and innovative smart fruit vending system. The key deliverables include:

- Hardware integration: Raspberry Pi 4, ESP32 microcontroller, Raspberry Pi camera, thermal printer, linear actuator, impulse sealer, touchscreen interface, load cell, and other sensors.
- Software development: Python libraries for image processing, object detection with YOLOv8, hardware interaction, UI development, and data management.
- Fruit recognition: Accurate fruit identification using a pre-trained YOLOv8 model.
- Automated weighing and pricing: Integration with a load cell for calculating fruit weight and generating price based on pre-defined formulas.
- QR code generation: Creation of QR codes containing product information and price for scan-and-pay options.
- User interface: A user-friendly touchscreen interface for selecting fruits, viewing weights and prices, and initiating purchase.

6.2 Challenges and Overcoming Strategies

Several challenges were encountered during project development, the following list contains some of them

- Hardware integration: Interfacing various hardware components with different communication protocols required thorough understanding of their specifications and implementation of appropriate libraries.

- Real-time performance: Balancing image processing speed with accuracy on the resource-constrained Raspberry Pi platform posed optimization challenges.

These challenges were overcome through the following strategies:

- Data augmentation: Techniques like image flipping, rotation, and color jittering were employed to artificially expand the training dataset and improve model generalizability.
- Modular software design: Dividing the software into independent modules facilitated debugging and independent optimization of each component.
- Performance optimization: Techniques like model quantization and code profiling were used to achieve real-time processing while maintaining accuracy.

6.3 Interesting Decisions

The project involved several interesting decisions that significantly impacted the final design:

- Choice of object detection model: Opting for YOLOv8 over other models due to its lightweight architecture and efficient real-time performance on the Raspberry Pi platform.
- Prioritization of user experience: Focusing on a user-friendly touchscreen interface with clear visuals and intuitive interactions enhanced customer satisfaction.
- Open-source software utilization: Leveraging readily available Python libraries and open-source resources accelerated development and facilitated future modifications.

These decisions contributed to the overall success of the project, demonstrating the importance of careful consideration and trade-offs during development.

6.4 Future Work

This project represents a solid foundation for further development and future releases. Potential features for future iterations include:

- Payment integration: Implementing various payment options like mobile wallets, NFC payments, or direct debit for seamless purchasing experiences.

- Product database expansion: Adding more fruits and vegetables to the system to cater to a wider range of customer preferences.
- Inventory management: Integrating a real-time inventory management system to track fruit availability and restock levels automatically.
- Automated bagging: Implementing a robotic arm or conveyor system for automated bagging of fruits after purchase.
- Data analytics and personalization: Utilizing collected data to analyze customer preferences and offer personalized recommendations or targeted promotions.

By incorporating these features in future releases, the smart cashier system can become even more versatile, convenient, and customer-centric.

List Of figures

FIGURE 1 GENERAL BLOCK DIAGRAM.....	8
FIGURE 2 FRUIT RECOGNITION ON A RASPBERRY PI [5]	11
FIGURE 3 SMART POS SYSTEM CASH REGISTER [6].....	12
FIGURE 4 LOAD CELL SENSOR [8].....	14
FIGURE 5 LCD MODULE [9]	15
FIGURE 6 ESP32.....	16
FIGURE 7 RASPBERRY PI 4.....	17
FIGURE 8 RASPBERRY PI CAMERA V2	18
FIGURE 9 HEATSINK CASE FOR RASPBERRY PI4	18
FIGURE 10 THERMAL PRINTER	19
FIGURE 11 IMPULSE SEALER [16]	20
FIGURE 12 LINEAR ACTUATOR [18].....	20
FIGURE 13 SYSTEM BLOCK DIAGRAM.....	22
FIGURE 14 SYSTEM FLOW CHART	23
FIGURE 15 SCHEMATIC DESIGN	26
FIGURE 16 SYSTEM EXTERIOR DESIGN	27
FIGURE 17 SYSTEM COMPONENTS	29
FIGURE 18 RASPBERRY PI4 PINOUT [23].....	29
FIGURE 19 ESP32 PINOUT [24].....	30
FIGURE 20 RASPBERRY PI4 CIRCUIT	30
FIGURE 21 ESP32 CIRCUIT	32
FIGURE 22 VNC CONFIGURATION.....	33
FIGURE 23 RASPBERRY PI SETTINGS	34
FIGURE 24 ENABLING RASPBERRY PI CAMERA STEPS	34
FIGURE 25 CAPTURE AN IMAGE COMMAND.....	35
FIGURE 26 TOUCHSCREEN CALIBRATION [26]	36
FIGURE 27 USED LIBRARIES	38
FIGURE 28 DATA SET PRUNING	39
FIGURE 29 GOOGLE COLAB NOTEBOOK	40
FIGURE 30 LOAD AND PROCESS THE PRICES FILE	42
FIGURE 31 LOAD YOLO MODEL.....	42
FIGURE 32 DETECT THE CLASS OF FRUIT	43
FIGURE 33 READ WEIGHT VALUE.....	43
FIGURE 34 SETUP GPIO PINS.....	44
FIGURE 35 THERMAL PACKAGING CODE	44
FIGURE 36 GENERATE QR CODE	45
FIGURE 37 READ THE QR CODE AND PRINT IT	46
FIGURE 38 LINEAR ACTUATOR AND IMPULSE TESTING	48
FIGURE 39 TOUCH SCREEN TESTING	49
FIGURE 40 PRECISION-RECALL (PR) CURVE	50
FIGURE 41 F1 SCORE CURVE	51
FIGURE 42 CONFUSION MATRIX	52
FIGURE 43 MODEL VALIDATION	52
FIGURE 44 QR CODE CONTENT.....	53
FIGURE 45 FUNCTIONAL TESTING.....	54
FIGURE 46 UI COMPONENTS TESTING	54

List Of Tables

TABLE 1 LOAD CELL SENSOR DATA SHEET.....	14
TABLE 2 COMPARSION BETWEEN MICROCONTROLLERS [7]	16
TABLE 3 RASPBERRY PI4 COMPONENT WIRING	31
TABLE 4 ESP32 COMPONENT WIRING	32
TABLE 5 YOLOV8 MODELS [20]	39
TABLE 6 WEIGHING SYSTEM TESTING.....	47

References

- [1] R. C. & .W. R. E. Gonzalez تأليف ،*Digital image processing* ،Pearson Education .2018 ،
- [2] A. Prasad ،"Composites for Packaging Application .[متصل]"،Available: https://www.researchgate.net/publication/357015952_Biodegradable_Composites_for_Packaging_Application.
- [3] "How to Grocery Shop With A Barcode Scanner .[متصل]"،Available: <https://www.outofmilk.com/ideas/barcode-scanner-app-usage/>
- [4] "Fruit Recognition on a Raspberry Pi .[متصل]"،Available: <https://www3.hs-albsig.de/wordpress/point2pointmotion/2020/03/26/fruit-recognition-on-a-raspberry-pi/>
- [5] "Fruit Recognition on a Raspberry Pi .[متصل]"،Available: <https://www3.hs-albsig.de/wordpress/point2pointmotion/2020/03/26/fruit-recognition-on-a-raspberry-pi/>
- [6] "ZHONGJI Smart POS System Cash Register with Weighing Checkout Scale .[متصل]"، Available: <https://www.amazon.com/ZHONGJI-Register-Weighing-Checkout-SET03-30LB/dp/B08CGT1NRW?th=1>.
- [7] "What Is A Load Cell And How Does It Work?," [Online]. Available: <https://www.flintec.com/weight-sensors/load-cells/what-is-a-load-cell>.
- [8] .[متصل]Available: <https://www.gotronic.fr/pj-460.pdf>.
- [9] .[متصل]Available: https://www.waveshare.com/wiki/7inch_HDMI_LCD.
- [10] "Is ESP32 Better than Arduino .[متصل]"،Available: <https://linuxhint.com/esp32-vs-arduino/>
- [11] "Raspberry Pi 4 .[متصل]"،Available: <https://www.raspberrypi.com/products/raspberry-pi-4-model-b/>
- [12] "Raspberry Pi Camera Module 2 .[متصل]"،Available: <https://www.raspberrypi.com/products/camera-module-v2/>
- [13] K. Douglas ،"Raspberry Pi 4 Heatsink: Placement, Size & ,More .[متصل]"،Available: <https://all3dp.com/2/raspberry-pi-4-heatsink-placement/>
- [14] "Interfacing Thermal Printer with Raspberry Pi to Print Text, Images, Barcodes and QR Codes .[متصل]"،Available: <https://circuitdigest.com/microcontroller-projects/thermal-printer-interfacing-with-raspberry-pi-zero-to-print-text-images-and-bar-codes>.
- [15] .[متصل]Available: <https://www.pack-secure.com/what-is-an-impulse-sealer.html>.

- [16] .[متصل]Available: <https://www.amazon.com/Impulse-Sealing-Machine-Plastic-Replace/dp/B07RRQBKQC>.
- [17] "What Is An Electric Linear Actuator .[متصل]"،Available: <https://www.timotion.com/en/news-and-articles/part-1-what-is-an-electric-linear-actuator-and-how-to-choose-it>.
- [18] .[متصل]Available: <https://www.amazon.com/Electric-Actuator-Feedback-Standing-Hospital/dp/B081CW4D1Z>.
- [19] R. A. Alan Faisandier, "Detailed Design Definition," [Online]. Available: https://sebokwiki.org/wiki/Detailed_Design_Definition.
- [20] [Online]. Available: <https://tfhub.dev/tensorflow/lite-model/efficientdet/lite0/detection/metadata/1>.
- [21] "Fruits and Vegetables Image Recognition Dataset .[متصل]"،Available: <https://www.kaggle.com/datasets/kritikseth/fruit-and-vegetable-image-recognition/data>.
- [22] "What is barcode and how is it made .[متصل]"،Available: <https://www.jagranjosh.com/general-knowledge/what-is-barcode-and-how-is-it-made-1522059273-1>.
- [23] .[متصل]Available: <https://www.raspberrypi.com/documentation/computers/raspberrypi.html>.
- [24] .[متصل]Available: <https://mischianti.org/esp32-devkitc-v4-high-resolution-pinout-and-specs/>
- [25] .[متصل]Available: <https://projects.raspberrypi.org/en/projects/getting-started-with-picamera>.
- [26] 7"inch HDMI LCD،" waveshare .[متصل]"،Available: https://www.waveshare.com/wiki/7inch_HDMI_LCD#Touch_calibration.
- [27] .[متصل]Available: <https://github.com/ultralytics/ultralytics>.
- [28] "Train and Export the model .[متصل]"،Available: <https://colab.research.google.com/drive/1AemZ4dbT3GGXMXhihi65j8AQxAX1xa0d#scrollTo=P7CnJTeCc-Qd>.
- [29] .[متصل]Available: <https://byjus.com/maths/absolute-and-relative-error/>
- [30] .[متصل]Available: <https://www.geeksforgeeks.org/precision-recall-curve-ml/>
- [31] .[متصل]Available: <https://docs.ultralytics.com/guides/yolo-performance-metrics/>
- [32] F. Femling, A. Olsson and F. Alonso-Fernandez, "Fruit and Vegetable Identification Using Machine Learning for Retail Applications," [Online]. Available:

https://www.researchgate.net/publication/328475022_Fruit_and_Vegetable_Identification_Using_Machine_Learning_for_Retail_Applications.

- [33] A. U. Alam, P. Rathi, H. Beshai, G. K. Sarabha and M. J. Deen, "Fruit Quality Monitoring with Smart Packaging," [Online]. Available: <https://www.mdpi.com/1424-8220/21/4/1509>.
- [34] Chattopadhyay, D., Chattopadhyay, S & ,Jana, D. (2019). Handbook of Research on Emerging Trends in Electrical and Computer Engineering. IGI Global ..
- [35] Hoffman, A. J. (2019). Applied Research Methods for Business and Management: An Introduction to Research and Methodology. SAGE Publications ..
- [36] "Rousseau, D. (2020). Computer Vision and Machine Learning for Fruit Detection and Localization. In Proceedings of the European Conference on Computer Vision Workshops (ECCVW).".
- [37] "UPC-A BARCODES .[متصل]"،Available: <https://www.cognex.com/resources/symbologies/1-d-linear-barcodes/upc-a-barcodes>.
- [38] .[متصل]Available: <https://www.geeksforgeeks.org/precision-recall-curve-ml/>